



# UNITED STATES PATENT AND TRADEMARK OFFICE

18W  
UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/925,877	08/09/2001	Catherine Elizabeth Korfanty Sheets	4676	6178

7590 06/09/2004

Carnes, Cona & Dixon  
Innovation Park  
1673 West Paul Dirac Drive  
Tallahassee, FL 32310-3763

EXAMINER
----------

LEROUX, ETIENNE PIERRE

ART UNIT	PAPER NUMBER
----------	--------------

2171

DATE MAILED: 06/09/2004 5

Please find below and/or attached an Office communication concerning this application or proceeding.

RECEIVED  
FEB 24 2005  
Technology Center 2100

## Office Action Summary

Application No.

09/925,877

Applicant(s)

SHEETS, CATHERINE ELIZABETH  
KORFANTY

Examiner

Etienne P LeRoux

Art Unit

2171

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

### Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

### Status

- 1) ☐ Responsive to communication(s) filed on \_\_\_\_.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

### Disposition of Claims

- 4) ☒ Claim(s) 1-5 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_ is/are withdrawn from consideration.
- 5) ☐ Claim(s) \_\_\_\_ is/are allowed.
- 6) ☒ Claim(s) 1-5 is/are rejected.
- 7) ☐ Claim(s) \_\_\_\_ is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_ are subject to restriction and/or election requirement.

### Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 09 August 2001 is/are: a) ☒ accepted or b) ☐ objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

### Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some \* c) ☐ None of:
- ☐ Certified copies of the priority documents have been received.
  - ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_.
  - ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

\* See the attached detailed Office action for a list of the certified copies not received.

### Attachment(s)

- 1) ☒ Notice of References Cited (PTO-892)
- 2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)  
Paper No(s)/Mail Date 4.
- 4) ☐ Interview Summary (PTO-413)  
Paper No(s)/Mail Date. \_\_\_\_.
- 5) ☐ Notice of Informal Patent Application (PTO-152)
- 6) ☐ Other: \_\_\_\_.

***Specification:***

**ABSTRACT:**

Applicant is reminded of the proper language and format for an abstract of the disclosure.

The abstract should be in narrative form and generally limited to a single paragraph on a separate sheet within the range of 50 to 150 words. It is important that the abstract not exceed 150 words in length since the space provided for the abstract on the computer tape used by the printer is limited. The form and legal phraseology often used in patent claims, such as "means" and "said," should be avoided. The abstract should describe the disclosure sufficiently to assist readers in deciding whether there is a need for consulting the full patent text for details.

The language should be clear and concise and should not repeat information given in the title. It should avoid using phrases which can be implied, such as, "The disclosure concerns," "The disclosure defined by this invention," "The disclosure describes," etc.

***Claim Rejections - 35 USC § 102***

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

Claims 1-5 are rejected under 35 U.S.C. 102(b) as being anticipated by US Pat No 6,016,478 issued to Zhang et al (hereafter Zhang).

Claim 1:

Zhang discloses a coded system for use on a computer and including a journal software package, said coded system comprising:

Art Unit: 2171

- at least one module; a plurality of prompts in the form of questions and an entering device for enabling a user to answer said questions [Figs 5A-G and col 11, line 5 - col 12, line 10]
- said plurality of prompts is directed to a specific category and will aid an individual to organize entered data [Figs 5A-G and col 11, line 5 - col 12, line 10]
- said questions provide input of needs, relationship, demographically information, preferences, strengths and pertinent information of said individual and said questions are to be answered by said individual [Figs 5A-G and col 11, line 5 - col 12, line 10]
- said at least one module is computer code and is coupled to an existing journal software package [Figs 5A-G and col 11, line 5 - col 12, line 10]
- an entering device for entering answers from said questions into said computer and provides data for a data base and said data base is used by said computer code of said at least one module [Figs 5A-G and col 11, line 5 - col 12, line 10]
- said at least one module interprets, processes and analyzes said data for organizing said data and links to said journal for enabling appropriate output [

Claim 2:

Zhang discloses wherein at least two modules are provided and each module includes a separate set of prompts containing questions geared to a particular category, each module being a separate computer code and a user can select which module to active [Fig 5C].

Claim 3:



Zhang discloses wherein a security system is included for enabling a user to correctly input in a code for accessing said coded system and said security system and said at least one module form a controlling station [Fig 5C]

Claim 4:

Zhang discloses wherein each of said at least one module includes prompts, which will enable advance-planning capability [col 11, lines 5-15]

Claim 5:

Zhang discloses wherein each of said at least one module includes the capability of calculating a specific date for alerting a user to do a specific task prior to a specific event [Fig 5H].

***Conclusion***

The prior art made of record and not relied upon is considered pertinent to applicant's disclosure:

- 1) US Pat No. 6,369,840 issued to Barnett et al discloses generating and displaying a calendar containing user-selected events from user-selected categories,
- 2) US Pat No. 6,466,236 issued to Pivowar et al discloses a portable handheld personal digital assistant for simultaneously displaying multiple calendars.

Art Unit: 2171

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Etienne LeRoux whose telephone number is (703) 305-0620.

The examiner can normally be reached on Monday – Friday from 8:00 AM to 4:30 PM.

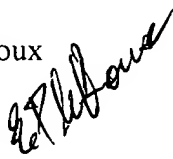
If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Safet Metjahic, can be reached on (703) 308-1436.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-3900.

Patent related correspondence can be forwarded via the following FAX number (703) 872-9306

Etienne LeRoux

June 4, 2004

A handwritten signature in black ink, appearing to read 'Etienne LeRoux', is written over the printed name and date.

<b>Notice of References Cited</b>	Application/Control No. 09/925,877	Applicant(s)/Patent Under Reexamination SHEETS, CATHERINE ELIZABE	
	Examiner Etienne P LeRoux	Art Unit 2171	Page 1 of 1

**U.S. PATENT DOCUMENTS**

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
	A	US-6,016,478	01-2000	Zhang et al.	705/9
	B	US-6,466,236	10-2002	Pivowar et al.	345/835
	C	US-6,369,840	04-2002	Barnett et al.	345/853
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

**FOREIGN PATENT DOCUMENTS**

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

**NON-PATENT DOCUMENTS**

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	
	V	
	W	
	X	

\*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)  
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

Form PTO -1449

U.S. DEPARTMENT OF COMMERCE  
PATENT AND TRADEMARK OFFICEINFORMATION DISCLOSURE  
STATEMENT BY APPLICANT
 Jc997 U.S. PTO  
 09/925877  
 08/09/01

<b>ATTY. DOCKET NO.</b> 4676	<b>SERIAL NUMBER</b>
<b>APPLICANTS</b> Catherine Elizabeth Korfanty Sheets	
<b>FILING DATE</b>	<b>GROUP</b> 2171

## U.S. PATENT DOCUMENTS

Examiner Initial	Document Number	Date	Name	Class	Sub-class	Filing Date if appropriate
ERR	6026410	2/5/00	Allen et al	707	104	2/10/97
	60183431	2/25/00	Wang et al	343	356	9/27/96
	558844	12/24/96	Smith et al	395	500	7/30/93

## FOREIGN PATENT DOCUMENTS

Examiner Initial	Document Number	Date	Country	Class	Sub-class	Translation Yes	No

## OTHER DOCUMENTS (Including author, title, date, pertinent pages, etc.)

Examiner Initial	Document

Examiner <i>EP Lehouc</i>	Date considered 6/7/04
Examiner: Initial if citation considered, whether or not citation is in conformance with MPEP 609; Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.	



US006016478A

# United States Patent [19]

Zhang et al.

[11] Patent Number: 6,016,478  
[45] Date of Patent: Jan. 18, 2000

[54] **SCHEDULING SYSTEM WITH METHODS FOR PEER-TO-PEER SCHEDULING OF REMOTE USERS**

[75] Inventors: Qili Zhang, Scotts Valley; Jin T. Teh, Fremont; Philippe Richard Kahn, Scotts Valley, all of Calif.

[73] Assignee: Starfish Software, Inc., Scotts Valley, Calif.

[21] Appl. No.: 08/693,677

[22] Filed: Aug. 13, 1996

[51] Int. Cl.<sup>7</sup> ..... B42D 5/04

[52] U.S. Cl. .... 705/9; 705/8; 395/200.32

[58] Field of Search ..... 705/9, 8; 395/200.32, 395/200.33, 200.37, 200.47, 200.57, 200.61, 200.68

[56] **References Cited**

**U.S. PATENT DOCUMENTS**

4,769,796	9/1988	Levine	368/29
4,807,155	2/1989	Cree et al.	345/329
4,866,611	9/1989	Cree et al.	364/300
4,881,179	11/1989	Vincent	364/518
4,977,520	12/1990	McGaughey, III et al.	364/521
5,113,380	5/1992	Levine	368/10
5,428,784	6/1995	Cahill, Jr.	395/200.36
5,493,692	2/1996	Theimer et al.	455/26.1
5,519,606	5/1996	Frid-Nielsen et al.	364/401

**OTHER PUBLICATIONS**

Mallett, M., "Sorting Out Schedules," BYTE, Dec. 1991, pp. 263, 264, 266, 268.

Microsoft Windows User's guide for the Windows Graphical Environment—Version 3.0, Copyright 1985–1990, pp. 50–53 and 370–382.

Mefford, M., "Track your time and appointments with schedule.com (includes related articles on customizing schedule and schedule command)," PC Magazine, v9, n6, Mar. 27, 1990, Dialog file 47, Accession No. 03386823, p. 303(7).

Kurkowski, C., "In sync; CrossWind's Synchronize software coordinates users' schedules and company resources for efficient project management. (CrossWind Technologies, Inc.'s project management software)," DG Review, v12, n7, Jan. 1992, Dialog file 275, Accession No. 01512963, p. 20(3).

Gunnerson, G., "Staying in Sync, at Home and on the Road," Network Computing, n210, 104, Date 911001, 1991, Dialog file 647, Accession No. 00605915, 3 pages.

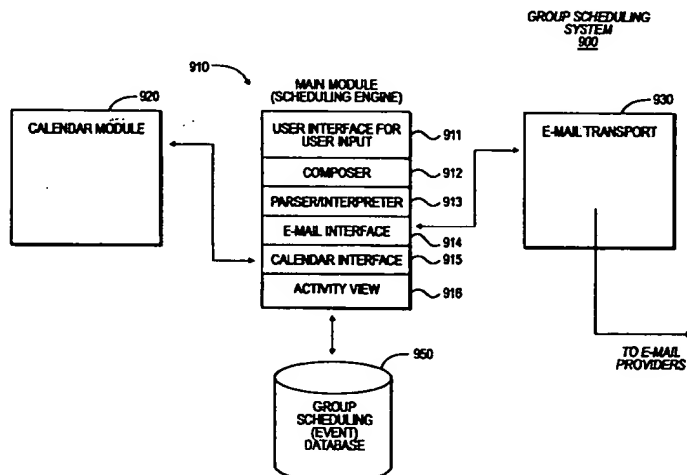
Primary Examiner—Thomas R. Peeso

Attorney, Agent, or Firm—John A. Smart

[57] **ABSTRACT**

An electronic Personal Information Manager (PIM) including a peer-to-peer group scheduling/calendar system is described. The group scheduling/calendar system provides methods for peer-to-peer group scheduling among users, including those users who only have simple e-mail support (i.e., do not have access to the group scheduling/calendar system itself). If a user is able to receive and respond to e-mail, he or she is able to participate in group scheduling in an automated fashion. Under user control, the system generates a scheduling invitation incorporating different formats. Each format includes, in order of increasing content richness, a simple text embedded scheduling invitation, an HTML (Hypertext Markup Language) form embedded scheduling invitation, and a proprietary binary "MIME" (Multipurpose Internet Mail Extensions) scheduling invitation. Each format is designed to transfer the highest degree of information content which a particular target client type can handle. A recipient of the scheduling message employs the messaging format best suited for his or her environment. Regardless of which format the recipient employs, the group scheduling system processes the reply message automatically, with the appropriate information automatically included in the user's group scheduling calendar. The system supports different levels of participation of various individuals throughout various stages of group scheduling, despite the fact that some of the individuals who need to participate might use other proprietary software and reside in other time zones.

36 Claims, 36 Drawing Sheets



# SCHEDULING SYSTEM WITH METHODS FOR PEER-TO-PEER SCHEDULING OF REMOTE USERS

## COPYRIGHT NOTICE

A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

## BACKGROUND OF THE INVENTION

The present invention relates generally to the area of information processing and, more particularly, apparatus and methods for managing and scheduling time-based information for multiple individuals located at different locations.

Successful management of one's time is a goal that every successful professional must achieve. One's business day may be swept away in a deluge of meetings and appointments, all of which must be somehow managed. An attempt to manage this task on paper, such as with a simple wall calendar, is unworkable for all but the simplest of schedules. More likely, such unsophisticated aids to managing one's time will lead to scheduling conflicts, missed appointments, botched deadlines, and angry clients.

Oftentimes, it is necessary to schedule a group of people for an event, such as a meeting. This is the problem of "group scheduling"—that is, notifying a group of people that a certain event is going to happen and receiving confirmation from members of the group whether each can participate. Conventionally, "group scheduling" has been largely limited to scheduling events for users within a particular "work group." Typically, a "work group" has comprised those users connected together through a local area network (LAN). Alternatively, a "work group" can be extended to users who can receive messages. In this extended group, however, manual processing on the part of the user is typically required. For instance, for a user who connects from a remote location, the user is required to manually process messages received to manually update the calendaring product to update one's scheduling status information. This leads to two disjointed activities for the user: (1) retrieving messages and (2) entering/processing scheduling information.

With the ever increasing importance of the Internet, work groups are no longer confined to local area networks, or even wide area networks (WANs). Instead, people are connected together via electronic mail or "e-mail." At the same time, however, users have become accustomed to the ease which automatic scheduling systems provide, such as those which operate within a proprietary environment (e.g., Novell Groupwise® operating on a Netware® local area network). If users are not connected to the same proprietary system (e.g., Novell Groupwise), then the users must resort to a manual scheduling process. Here, the job typically falls to a secretary or administrative assistant who must contact each proposed participant individually, for determining his or her availability. Typically, the person charges with scheduling the event must "negotiate" with the proposed participants for reaching a time when the meeting can finally happen. The process is still not complete, however. A confirmation message must be sent to all proposed participants for confirming the final time.

What is really needed are system and methods which permit any user to schedule appointments with a group of

other users in an automated fashion, but without requiring the users of the group to employ or be connected to a particular proprietary system. In particular, such a system should allow a user to initiate a message to invite a group of people to a meeting (i.e., to schedule a meeting). Those individuals should, if they are able to receive electronic mail, be able to participate in the group scheduling. Here, the recipients need not have access to any particular proprietary software for participating in the group scheduling. Instead, each participant need only be able to receive and send e-mail (which can be done using a wide variety of products, from numerous vendors.) The present invention fulfills this and other needs.

## SUMMARY OF THE INVENTION

The present invention recognizes a user needs flexibility in choosing how appointments, events, and other time-based data are entered and managed, despite the fact that other individuals who need to participate might use other proprietary software and reside in other time zones. According to the present invention, therefore, an electronic group scheduling/calendar system is provided with methods for peer-to-peer group scheduling among users, including those users who only have simple e-mail support (i.e., do not have access to the group scheduling/calendar system itself), or even have no e-mail support.

According to the present invention, if a user is able to receive and respond to e-mail, he or she should be able to participate in group scheduling in an automated fashion. In particular, the present invention allows a user to undertake group scheduling with other "remote" users located at different locations (including those with different time zones), regardless of what particular platform or software applications each of the other users is employing. In contrast to prior art scheduling approaches which required all users to operate the same proprietary scheduling software, the present invention instead requires only one user of a work-group to have the group scheduling software which the present invention is embodied. Every other user need only be able to send and receive e-mail messages—using any one of the wide variety of e-mail packages which are available—in order to be automatically tracked by the system. Still further, the present invention includes facilities for accommodating even those users who cannot receive e-mail.

"E-mail" itself is a messaging-based approach which is exploited by the present invention for communicating with all users, regardless of which proprietary system a given user is employing. The present invention implements a messaging subsystem or exchange which provide scheduling primitives (e.g., "accept" and "decline" message types) for supporting the basic functionality of group scheduling, even for generic e-mail clients—that is, a client which does not share nor is even aware of the group scheduling software subsystem provided by the local client of the system of the present invention. This include "dumb" remote clients which simply have no knowledge or understanding of the scheduling format supported by the group scheduling local client.

In typical operation, for instance, group scheduling begins with a user scheduling an event, that is, sending to desired participants an initial meeting message or "invitation" for scheduling the event. The event itself can be any type of event, including those with a duration (e.g., meetings and appointments); "resources" (e.g., conference rooms, projectors, and the like) can also be scheduled. The recipient user(s) can receive the message across a variety of different software platforms or e-mail solutions.

The message itself comprises an identifier, such as <ISK> (or other unique), together with a scheduling invitation in different formats. The different formats include, in order of increasing content richness, a simple text embedded scheduling invitation, an HTML (Hypertext Markup Language) form embedded scheduling invitation, and a proprietary binary "MIME" (Multipurpose Internet Mail Extensions) scheduling invitation. Each format is designed to transfer the highest degree of information content which a particular target client type can handle. A remote client having the scheduling system of the present invention can recognize from the identifier that the message is from another proprietary client. Accordingly, the client can process the binary "MIME" scheduling invitation natively. This message is a vendor-specific message, allowing direct, proprietary communication between proprietary clients with the richest possible information content.

Since other clients have no knowledge of the scheduling system, they simply ignore the identifier and the binary attachment. To allow the local client to communicate with these recipient, non-proprietary interclient communication formats are employed. For instance, a remote client with browser capability can employ the HTML form embedded scheduling invitation, which can be appropriately processed by its Web browser (e.g., Microsoft Explorer or Netscape Navigator); this represents the richest content that this client type can handle. The embedded HTML form (i.e., Web page) can easily be viewed by the Web browser as an input form having input fields corresponding to the information requested for scheduling the event. For instance, the form may include text or input fields for subject, time, event, and the like. Additionally, the form can include screen buttons for allowing the recipient user to "accept" or "decline" the invitation. At the conclusion of completing the input form, the recipient user can select "submit" for returning the input information back to the initiator. Here, the browser of the recipient generates a reply message, in a manner similar to that done today for on-line Internet registration. Upon receipt of the reply, the originating client can identify, decode, and process the reply appropriately.

In the event that the recipient client is a simple e-mail client, the recipient client employs the simple messaging format incorporated into the scheduling message. Here, the recipient client can view a plain text e-mail message to which the user can respond (e.g., "accept" or "decline"). In the recipient client's reply, the client includes the "body" of the initial invitation message. As is common in e-mail communication, the reply message can itself easily incorporate the contents of the original message. By simply including the initiator's original message when a non-SK client replies, the non-SK client is incorporating information which facilitates processing of the reply by the SK local client. For instance, the reply includes the "ISK" signature which is embedded within the subject line. Other information includes the e-mail address of the recipient as well as the recipient's response (e.g., "accept") using delimited key words. Upon receiving the reply, the initiator can recognize that the response (e.g., an "accept" message or a "decline" message) corresponds to a particular invitation send out, thus facilitating decoding and processing of the message. In this manner, the response includes sufficient scheduling information embedded within it to allow the initiator client to appropriately decode the response, despite the fact that the responding recipient is a simple e-mail client without browser capability.

Regardless of which format the recipient employs, the group scheduling system of the present invention can pro-

cess the reply message automatically, including entering the appropriate information in the user's group scheduling calendar. In this manner, the system of the present invention can support different levels of participation of various users (none of which is required to have the system), throughout various stages of group scheduling.

#### BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1A is a block diagram of a computer system in which the present invention may be embodied.

FIG. 1B is a block diagram of a software system of the present invention for controlling the operation of the system of FIG. 1A.

FIG. 2 is a bitmap screenshot illustrating a user interface of a Personal Information Manager which embodies the present invention.

FIGS. 3A-C illustrate interclient communication which employs a message incorporating multi-format information embedded within.

FIG. 4 is a bitmap screenshot illustrating a "Desktop" interface provided by the system.

FIGS. 5A-I are bitmap screenshots illustrating a Scheduling Wizard interface provided by the system for scheduling events.

FIGS. 6A-C are bitmap screenshots illustrating a preferred interface provided by the system for processing incoming event invitations.

FIGS. 7A-C are bitmap screenshots illustrating a preferred interface provided by the system for processing incoming replies.

FIGS. 8A-G are bitmap screenshots illustrating a preferred interface provided by the system for scheduling use of resources.

FIG. 9 is a block diagram providing an overview of the internal architecture of a group scheduling system of the present invention.

FIG. 10 is a block diagram illustrating the general process of sending an event.

FIG. 11 is a block diagram illustrating the general process of receiving messages.

FIGS. 12A-C are bitmap screenshots illustrating receipt of an e-mail scheduling invitation by a recipient whose system represents a non-SK client without browser support.

FIG. 13 is a screenshot illustrating receipt of an e-mail scheduling invitation by a recipient whose system represents a non-SK client with browser support.

#### DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

The following description will focus on the presently preferred embodiment of the present invention, which is operative in an end-user application running under the Microsoft® Windows environment. The present invention, however, is not limited to any particular one application or any particular environment. Instead, those skilled in the art will find that the system and methods of the present invention may be advantageously applied to a variety of system and application software, including database management systems, wordprocessors, spreadsheets, and the like. Moreover, the present invention may be embodied on a variety of different platforms, including Macintosh, UNIX, NextStep, and the like. Therefore, the description of the exemplary embodiments which follows is for purposes of illustration and not limitation.

### System Hardware

The invention may be embodied on a computer system such as the system 100 FIG. 1A, which comprises a central processor 101, a main memory 102, an input/output controller 103, a keyboard 104, a pointing device 105 (e.g., mouse, track ball, pen device, or the like), a display or screen device 106, and a mass storage 107 (e.g., hard or fixed disk, removable floppy disk, optical disk, magneto-optical disk, or flash memory). Although not shown separately, a real-time system clock is included with the system 100, in a conventional manner. Processor 101 includes or is coupled to a cache memory 109 for storing frequently accessed information; memory 109 may be an on-chip cache or external cache (as shown). One or more input/output device(s) 108, such as a printing device or slide output device, are included in the system 100, as desired. As shown, the various components of the system 100 communicate through a system bus 110 or similar architecture. In a preferred embodiment, the system 100 includes an IBM PC-compatible personal computer, available from a variety of vendors (including IBM of Armonk, N.Y.). I/O device 108 may include a laser printer, such as an HP Laserjet printer, which is available from Hewlett-Packard of Palo Alto, Calif.

### System Software

#### A. Overview

Illustrated in FIG. 1B, a computer software system 120 is provided for directing the operation of the computer system 100. Software system 120, which is stored in system memory 102 and on storage (e.g., disk memory) 107, includes a kernel or operating system (OS) 140 and a windows shell 150. One or more application programs, such as client application software or "programs" 145 may be "loaded" (i.e., transferred from storage 107 into memory 102) for execution by the system 100.

System 120 includes a user interface (UI) 160, preferably a Graphical User Interface (GUI), for receiving user commands and data. These inputs, in turn, may be acted upon by the system 100 in accordance with instructions from operating module 140, windows 150, and/or client application module(s) 145. The UI 160 also serves to display the results of operation from the OS 140, windows 150, and application (s) 145, whereupon the user may supply additional inputs or terminate the session. OS 140 and windows 145 can be provided by Microsoft® Windows 95, by Microsoft® Windows NT, or by Microsoft® Windows 3.x (operating in conjunction with MS-DOS); these are available from Microsoft Corporation of Redmond, Wash. Although shown conceptually as a separate module, the UI is typically provided by interaction of the application modules with the windows shell, both operating under OS 140.

One application software comprises a Personal Information Management (PIM) System 125 which includes an Internet-based Group Scheduling Module 127 of the present invention. The Internet-based Group Scheduling Module 127 provides group scheduling among users connected to the Internet or to other commercial service providers (e.g., CompuServe). In an exemplary embodiment, PIM System 125 comprises Sidekick®, which is available from Starfish Software, Inc. of Scotts Valley, Calif. A general description of the operation of Sidekick® can be found in its accompanying user manual. Interface and methods provided by the group scheduling module of the present invention, in the exemplary embodiment of Sidekick®, will now be described in further detail.

### Peer-to-Peer Scheduling

#### A. General

According to the present invention, if a user is able to receive and respond to e-mail, he or she should be able to

participate in group scheduling. In particular, the present invention allows a user to undertake group scheduling with other "remote" users located at different locations (including those with different time zones), regardless of what particular platform or software applications each of the other users is employing. In contrast to prior art scheduling approaches which required all users to operate the same proprietary scheduling software, the present invention instead requires only one user of a workgroup to have the group scheduling software which the present invention is embodied. Every other user need only be able to send and receive e-mail messages—using any one of the wide variety of e-mail packages which are available. Still further, the present invention includes facilities for accommodating even those users who cannot receive e-mail.

"E-mail" itself is a messaging-based approach which is employed by the present invention for communicating with all users. The present invention implements a messaging subsystem or exchange which provide scheduling primitives—for example, "accept" and "decline" message types—for supporting the basic functionality of group scheduling. In typical operation, for instance, group scheduling begins with a user scheduling an event, that is, sending to desired participants an initial meeting message or "invitation" for scheduling the event. The event itself can be any type of event, including those with a duration (e.g., meetings and appointments); as described below, "resources" (e.g., conference rooms, projectors, and the like) are also scheduled. The recipient user(s), once having received the message, can undertake one of several actions. The user can accept or decline to participate, or the user can forward the message to one or more other users. When declining a proposed scheduling event, the user can propose another time for the event. When a user declines, he or she replies with a "decline" message; the message may include one or more alternative times proposed by the declining user. The decline message is processed upon its return, whereupon the system automatically updates the scheduling calendar. An accepting user, on the other hand, replies with an "accept" message. Again, the group scheduling system processes the message automatically, including entering the appropriate information in the group scheduling calendar. In this manner, the system of the present invention can support different levels of participation of various users (none of which is required to have the system), throughout various stages of group scheduling.

#### B. Functional Overview

FIG. 2 is a block diagram illustrating a functional overview of a group scheduling system 200 of the present invention, from the perspective of an end user. The system includes a local client 210 comprising front-end software, including a graphical user interface, for entering scheduling information and generating group scheduling messages. The client 210 connects to transport mechanism or messaging engine 220. This provides the mechanism whereby the client 210 can send messages to and receive messages from remote clients.

In a preferred embodiment, the messaging engine 220 supports Microsoft® Exchange or MAPI (Messaging Application Programming Interface), AOL (America On Line), and POP 3 (Internet "Post Office Protocol") messaging protocols. These widely-supported protocols can be employed for reaching remote clients having addresses (i.e., accounts) on the Internet, America On Line, Microsoft Network (MSN), as well as other systems which are accessible through gateways (e.g., CompuServe, which has an Internet gateway). To further support each messaging or



mail system, the client 210 includes an address book module 230, which comprises a separate address book for each messaging system. In an exemplary embodiment, for instance, the system provides support for AOL address book, Internet (Netscape) address book, and MAPI-exchange address book.

With the messaging engine connection, the client 210 is a "local" client which exchanges messages with various "remote" clients, such as client 240, client 250, and client 260. As shown, there are different types of remote clients. One type of remote client is one which shares the same software group scheduling subsystem as the local client 210 (i.e., it "understands" proprietary file formats of the local client 210). Such a client is shown at 240. Here, the local client 210 and the remote client 240 can of course communicate directly using the proprietary format understood by both, in the same manner as presently done by present-day proprietary group scheduling solutions.

According to the present invention, however, a remote client may also comprise a generic e-mail client—that is, a client which does not share nor is even aware of the group scheduling software subsystem provided by the local client of the present invention. These "dumb" remote clients, such as represented by clients 250 and 260, simply have no knowledge or understanding of the scheduling format supported by the local client 210. For clarity of the description which follows, the generic or non-proprietary clients are referred to as "non-SK" clients, for indicating that these clients do not have Sidekick®—the proprietary software which implements automated group scheduling with both proprietary and non-proprietary clients. Those clients which do have Sidekick® are referred to as "SK" clients.

With the ever-increasing use of the Internet and particular the rapid adoption of the World-wide Web ("Web") as a pervasive communication platform, an ever-increasing number of non-proprietary remote clients will have a standard browser, such as Netscape Navigator (available from Netscape of Mountain View, Calif.) or Microsoft Internet Explorer (available from Microsoft Corp. of Redmond, Wash.). As described in further detail below, the present invention may exploit this by using rich text messages, such as e-mail including one or more HTML (Hyper Text Markup Language) forms or "Web pages." Such a client, such as shown at 250 in FIG. 2, is identified as a remote non-SK client "with browser." A simple e-mail client, on the other hand, is identified as a remote non-SK client "without browser."

#### C. Interclient Communication

The local SK client communicates with other clients by employing its messaging engine or transport layer to send and receives electronic mail or messages. As illustrated in FIGS. 3A–C, each message itself is structured to facilitate an optimal communication protocol appropriate for the different types of clients (i.e., SK client, non-SK client with browser, or non-SK client without browser). This is achieved by incorporating into the message different formats of the scheduling message—one format for each particular client type.

To allow rapid identification of scheduling messages between SK clients themselves, an identifier or "signature" is incorporated into the message by tagging the standard "subject line" in each e-mail message. In an exemplary embodiment, the initiator of a message (i.e., local SK client) includes the following unique identifier in the subject line of <ISK> or, alternatively, <SIS>. The interpretation of this depends on the actual type of client the recipient is. Remote recipient clients which do support the proprietary format of

the local client (i.e., recipient SK remote clients) can recognize the identifier and readily identify the message as a scheduling message from another SK client. This allows all SK clients to easily identify scheduling messages among themselves, thus facilitating decoding and processing of messages. In the instance of one SK client communicating with another SK client, therefore, the recipient is a "smart" client with full knowledge and understanding of the type of message and how to interpret it.

One such SK-client-to-SK-client dialog is illustrated by interclient communication 300 in FIG. 3A. Local SK client sends e-mail message 310. The message comprises the <ISK> (or other unique) identifier 311 together with a scheduling invitation in different formats. The different formats include, in order of increasing content richness, a simple text embedded scheduling invitation 313, an HTML (Hypertext Markup Language) form embedded scheduling invitation 315, and a proprietary binary "MIME" (Multipurpose Internet Mail Extensions) scheduling invitation 317.

Each format is designed to transfer the highest degree of information content which a particular target client type can handle. In FIG. 3A, for instance, remote client 320 recognizes from the identifier 311 that the message 310 is from another SK client—SK local client 301. Accordingly, the remote SK client 320 processes the binary "MIME" scheduling invitation 317. This message is a SK-specific message, allowing direct, proprietary communication between SK clients with the richest possible information content. Description of the MIME format is well documented in the trade literature; see e.g., Kientzle, T., *MIME and Internet Mail*, Dr. Dobbs' Journal, September 1995, pp. 54–60 (with code listings pp. 104–110), the disclosure of which is hereby incorporated by reference.

Since non-SK clients have no knowledge of the scheduling system, they simply ignore the identifier and the SK binary attachment. To allow the SK local client to communicate with these non-SK recipient, the non-proprietary interclient communication formats 313, 315 are employed. As shown in FIG. 3B, for instance, remote non-SK client with browser 340 employs the HTML form embedded scheduling invitation 315, which can be appropriately processed by its Web browser (e.g., Microsoft Explorer or Netscape Navigator); this represents the richest content that this client type can handle.

The embedded HTML form (i.e., Web page) can easily be viewed by the Web browser as an input form having input fields corresponding to the information requested for scheduling the event. For instance, the form may include text or input fields for subject, time, event, and the like. Additionally, the form can include screen buttons for allowing the recipient user to "accept" or "decline" the invitation. At the conclusion of completing the input form, the recipient user can select "submit" for returning the input information back to the initiator. Here, the browser of the recipient generates a reply message, in a manner similar to that done today for on-line Internet registration. Upon receipt of the reply, the SK client 210 can identify, decode, and process the reply appropriately. Based on the action selected by the recipient (e.g., accept, decline, or the like), the SK client 210 can update its scheduling information in an appropriate manner.

In the event that the client is a non-SK client and does not have a browser, a simple messaging format is employed. As shown in FIG. 3C, remote non-SK client without browser 360 processes the simple text embedded scheduling invitation 313, as this represents the richest content that it can

handle. Here, the recipient client receives a plain text e-mail message to which it can respond (e.g., "accept" or "decline"). In the client's response, the client includes the "body" of the initial invitation message. As is common in e-mail communication, the reply message can itself easily incorporate the contents of the original message. By simply including the initiator's original message when a non-SK client replies, the non-SK client is incorporating information which facilitates processing of the reply by the SK local client. For instance, the reply includes the "ISK" signature which is embedded within the subject line. Other information includes the e-mail address of the recipient as well as the recipient's response (e.g., "accept") using delimited key words. Upon receiving the reply, the initiator can recognize that the response (e.g., an "accept" message or a "decline" message) corresponds to a particular invitation sent out, thus facilitating decoding and processing of the message. In this manner, the response includes sufficient scheduling information embedded within it to allow the initiator client to appropriately decode the response, despite the fact that the responding recipient is a non-SK client without a browser—a simple, generic e-mail client.

#### D. Resource Scheduling

When a meeting or an event is scheduled, often "resources" will be needed as well. For a board meeting, for instance, the board room needs to be available. The meeting might also require other resources, such as an overhead projector or a VCR. In accordance with the present invention, the scheduling system is extended to include these "resources." As with users, the resource scheduling is peer-to-peer—that is, it is shared among the users in a cooperative fashion.

A resource itself "owns" an e-mail account—that is, it is able to send and receive scheduling messages. In this manner, scheduling of the resource can be completely automated. Here, the resource is controlled by an SK client, which manages the calendar for the resource. When a request is received to schedule the resource, the SK client can check its calendar and respond whether the resource is available, all in an automated fashion. For the example of a board room meeting, the board room resource is, in effect, "invited" to the meeting. Through the SK client which is controlling the scheduling calendar for the board room, the board room can "reply." In a preferred embodiment, an immovable resource (e.g., "room" resource) cannot be scheduled to attend two meetings at the same time. A person, on the other hand, can be scheduled for two meetings at the same time; in that instance, the person decides which meeting to attend (i.e., which one is more important), or decides to attend certain portions of each meeting.

#### Preferred Interface and User Operation

##### A. General Interface

As shown in FIG. 4, the system provides a "Desktop" interface 400—that is, a personal information management interface which includes an electronic appointment calendar. The interface 400 includes a menu bar 410, for invoking menu items or commands. The interface itself is comprised of particular subregions. A first subregion 420 displays the current time and date. Below this region are a To Do region 430 and a Call region 440. The former lists To Do items which the user has included. The latter is a log of calls which the user wishes to track. Actual schedule events or appointments are displayed in the interface at region 450. By default, appointments are displayed in one's own local time. The interface also includes a quick lookup or viewport 460, for working with another view of the interface (e.g., "Cardfile" view). Finally, the interface includes Desktop icons 470, for quickly switching to other modules of the system.

The interface 400 provides different "views": Calendar view, Cardfile view, Write view, Expense view, and Activities view. The Cardfile view can be used as an address book in which the user stores names, addresses, e-mail addresses, phone numbers, and other information (e.g., record collection). The Cardfile works with the Calendar's Internet group scheduling to address event invitations, and also can look up phone numbers of incoming calls using Caller ID. The user can use the Cardfile with the Phone Dialer to dial calls, or merge card information into a letter using Quick Letter. The Write view provides a place to create and format documents, organized in folders and binders. The Expense view lets the user enter information from expense receipts, and organize expenses in folders. The user's expenses are printed in a finished expense report. The Activities view presents information about all the user's group scheduling events plus other Calendar activities: one's calls, To-Do items, and individual appointments for the period selected. The Activities view is employed to reply to group event invitations and view replies to group events the user has originated.

Of particular interest to the present invention is the Calendar view which, as shown in FIG. 4, displays the user's Calendar. Here, the user creates appointments and uses Internet-based messaging (or other electronic messaging service) to automatically handle invitations and replies for scheduling group events. The appointments adjust to the user's own local time zone, if the user travels with the computer when he or she switches to the new local time in an "EarthTime" module of the system. The EarthTime module tells the user the current time in eight locations around the globe, and provides other information such as time difference start and end of daylight saving time, and other facts about each city, for over 540 cities. Further description of the EarthTime module is provided in commonly-owned application Ser. No. 08/609,983, filed Feb. 29, 1996, which is hereby incorporated by reference. When desired, the user can print the Calendar in a variety of formats, including page sizes suited to the most popular day planners, such as Keith Clark, DayRunner, Day-Timers, and Franklin Planner.

##### B. Peer-to-Peer Scheduling Interface

###### 1. Overview

The Group Scheduling module 127 uses electronic messaging to communicate with others to schedule events and book resources. Because it supports numerous e-mail platforms, including the Internet, the module can provide group scheduling with users anywhere in the world. The system automatically invites the participants to an event the user schedules, and collects their replies for accepting or declining the invitation or for requesting that the user reschedule it. The event is automatically placed in the Calendar in the Calendar view, as well as the recipients' calendars if they accept. Additionally, the user can also automatically reserve resources for the event, such as conference facilities, projectors, and others. Most of the group scheduling occurs in the Activities View, where the user can see all group events in list form. Events the user schedules, or that the user accepts an invitation to, are also posted automatically in the user's Calendar.

To schedule an event, the user enters the details of the meeting and the list of people he or she wants to invite. The system then automatically notifies the recipients and collects their responses for the user. In addition to electronic messaging, the user can use fax or telephone for group scheduling, for people without e-mail accounts. If the user specifies a fax number for notification, the event invitation

is automatically sent using built-in fax software (e.g., Microsoft Fax) and the user's fax modem. Selecting phone notification, on the other hand, puts a reminder in the user's Calls list.

## 2. Scheduling Wizard

The system provides a convenient "Scheduling Wizard" to assist the user in the process of entering event information. It helps the user enter all the information about the event, and identify the participants he or she wants to invite. The user also can create a group event using a "Schedule an Event" dialog box. The user can include a message with the invitation, and the replies can include a message and a report on free time if a recipient cannot attend at the scheduled time. If the user needs to notify some people by fax or phone, he or she can enter their reply status manually.

The easiest way to enter group event information is using the Scheduling Wizard. To schedule an event in the Activities or Calendar view, the user selects Internet\Scheduling Wizard from the main menu, as shown at 501 in FIG. 5A. This action launches Scheduling Wizard 510, as shown in FIG. 5B. The wizard consists of a series of pages that prompt the user to enter the information for the event, such as participants, resources, agenda or notes, and options. The user enters the information in each page, and clicks "Next" to proceed to the next page. The user can go back at any time to earlier pages if he or she needs to change something. When the user has finished, the system displays an Event Summary. Here, the user can review his or her choices, and make changes, if necessary, by going back to previous pages. When the user is ready to schedule the event, the Wizard adds it to the user's queue of outgoing messages, and put it on the user's calendar. To use Internet scheduling, the user creates a cardfile that contains properly identified e-mail addresses for the people the user will be inviting. Alternatively, the user can use an existing Netscape Address Book or Microsoft Exchange Cardfile for addressing.

The wizard is used from start to finish as follows. First, the user enters an event type. As illustrated in FIG. 5C, the user clicks an event category at 511 and selects a particular subject—that is, an event type to schedule. When the user has completed each Wizard panel, the user clicks "Next" button 512 to continue. Alternatively, the user can click "Back" to return to an earlier panel. Next, in FIG. 5D, the user invokes the "Date and Time" page. Here, the user reviews the subject presently entered, at 521, and then enters a date and time for the event, at 522. The user can select the date by typing or by using the arrows. A calendar can be opened at this point (e.g., by selecting the large arrow). The user also enters the start and end times. Next, the user selects or types the city where the event is to occur, at 523. The time zone fills in automatically. After typing or selecting a place, the user advances the wizard to the next pane by selecting Next button 524.

The next pane is a "Participants" page, illustrated in FIG. 5E, which allows the user to select participants. At 531, the user chooses an Address Book or a mailing list; clicking "More" opens a different Address book. Now, the user clicks the folder next to each name, and clicks the notification method (i.e., e-mail, fax, or the like) for that participant. The user adds desired selections to the Participants list 533 or CC list 534, using selection buttons 535. Thereafter, the user moves to the next pane by selecting Next button 536. On the "Message" page, shown in FIG. 5F, the user types the agenda or notes, at 541. From buttons 542, the user can click a "Message" button to select a file containing a message. Alternatively, the user clicks "Attach URL" (Uniform Resource Locator, the address of a site on the World Wide

Web) to add an Internet address as part of the message information. Users receiving the URL can click on it to launch their Web browser and jump to the URL site. At 543, the user can select a file to send as an attachment to the meeting invitation. The user proceeds to the next pane by selecting Next button 544. The next pane or page is the "Resources" page, illustrated in FIG. 5G, where the user selects one or more resources from a list 551 of available resources. The user clicks the selection buttons 553 to add or remove items from the Event Resources list 552. By clicking Next button 554, the user proceeds to the next page.

The next page is the "Event options" page, illustrated in FIG. 5H. At 561, the user can select a check box to send a reminder as well as set an amount of advance notice. By checking "Page Me" or "Alert Me" and setting a time, the user can instruct the system to remind the user by page or alarm. The user selects the RSVP radio button, in Event Options box 562, to request a reply, or selects the FYI radio button to send an announcement only. By selecting RSVP (i.e., "please respond"), recipients can send back a reply whereupon the system automatically collects the information for the user. If the user chooses FYI (i.e., "For Your Information"), on the other hand, he or she is sending an announcement of the event, without requesting replies. This option is used, for example, for events such as a company meeting that will occur regardless of individual schedules. Additionally, in Event Options box 562, the user can instruct the system to enter this Event in the user's Contact Log. Upon checking the box, the event is entered in the Contact Log for each participant in the invitation list. Finally, the user can specify a type for the event: Public, Personal, or Confidential.

Upon selecting Next button 563, the user instructs the system to proceed to the "Schedule the Event" page, illustrated in FIG. 5I. Here, the user can review the selections, at 571, and go back to any previous page, if needed. Once satisfied with the selections, the user selects "Finish" button 572 to schedule the event. Alternatively, the user can cancel the input altogether by selecting "Cancel."

The user can also include people who do not have e-mail in the list of event invitees. Here, two options exist for addressing invitations: fax or telephone. If an invitation is directed to someone at their fax address, the system launches the fax software (e.g., Microsoft Fax) and faxes the invitation automatically. If the user selects someone's telephone number as their address, a reminder is added to the user's Calls list in the calendar to call that person. When the user hears from these people, he or she can enter their meeting reply by entering the appropriate status: Accept, Decline, Reschedule Request, or Delegate.

## 3. Incoming Event Invitations

When the user receives an invitation to an event, it appears in bold type in the Activities view 600, which is illustrated in FIG. 6A. An icon appears in the News column 623 to indicate a new event. The user will see the incoming event invitation in his or her e-mail in-box, marked with the signature (e.g., <ISK> for Internet Sidekick) in the subject. In typical operation, the user disregards this message in his or her e-mail in-box, as the system will automatically find and handle it for the user. Users who have other systems, on the other hand, can open the message like any other e-mail; it contains instructions on how to reply so the original sender's system (i.e., SK client) will be able to record the reply.

To get to the Activities view, the user click the Activities icon (or selects the corresponding menu command). After the user has clicked Group Events tab 611, status informa-

tion is displayed for group events in the three left-hand columns. Type column 621 displays an icon for the event type: (b 1) New event the user initiated; (2) New incoming event, and (3) Incoming event the user has opened. News column 623 displays an icon for all new items: (1) New event information; (2) Reply has arrived; and (3) Event has been canceled. Out-box column 625 indicates messages that are queued to be sent during the user's next e-mail flash session; it displays an icon indicating Messaging waiting to be sent. The user can select an Internet |Send/Receive Now command to send pending messages immediately. By clicking on a Detail item 627, the user can display further information in Details and Information pane 629.

To reply to an event invitation, the user reviews the event specifics in the Details and Information panes. Then, the user clicks a choice to reply from the buttons at the bottom of the window (e.g., "Accept," "Decline," "Reschedule," or the like). Alternatively, the user can right-click an event in the top pane, and choose from a Shortcut menu. In a preferred embodiment, the user's reply is sent only to the originator, not to others invited to the event.

Accept button 631 lets the user enter a short reply message (via Reply dialog 635 in FIG. 6B), and then sends the acceptance to the initiator confirming that the user will attend. The event is automatically added to the user's calendar. Decline button 633, on the other hand, sends a reply message to the initiator that the user will not be able to attend. The event is not added to the user's calendar. With the user's reply, the user can include a Free Time report showing the user's booked and unbooked time for the next 30 days (or other user-specified interval), for assisting the meeting originator in rescheduling. Included in the Free Time report is an indication of all the times for the next 30 days when the user has a scheduled event, and all other (free) times. In a preferred embodiment, no information about specific events on the user's calendar is sent, beyond the indication that the time is booked. The user checks box 637 to include the report.

In addition to accepting or declining, the recipient user can request rescheduling of the event. Here, the user selects Reschedule Request button, such as button 635 in FIG. 6A. In response, the system displays Reschedule Request dialog 640, as shown in FIG. 6C. This option sends a reply requesting that the event be moved to another time or date. In the dialog, the user can enter one or more alternate times for the event, as well as include a short message. As with the decline response, the user has the option of including a Free Time report to assist the meeting originator in rescheduling. The Minutes button 634 is used to send notes or minutes to all participants. This launches a "Compose the Minutes" window where the user can write a message or attach a file (by clicking a Browse button); the minutes are automatically distributed to the participants.

Besides accepting and declining, the user can "delegate" the event by choosing the option from the menu. The option opens a Delegate To dialog box (not shown), where the user can specify a person to forward the invitation to. Here, the meeting originator is notified that the user has delegated the event to someone else.

#### 4. Replying to Incoming Event Invitations for Non-SK Clients

Users who are non-SK client users (i.e., do not have Internet Sidekick installed) can still reply to incoming invitations in a way that is automatically recorded by the event originator's. All event invitations, whether the recipient is a non-SK client user or an SK client user, appear in the user's e-mail in-box. Users who are SK client users can

ignore the messages, since the system will read and process these automatically.

Non-SK client users can open the invitation like any other e-mail item and read the contents, which are stored in formatted text. The body of the message includes instructions that explain briefly how a non-SK client user can reply. To reply to an invitation using conventional e-mail software, the non-SK client user proceeds as follows. The user opens the invitation in his or her e-mail in-box, like any other message. Using the user's own e-mail software (from any vendor), the user "replies" to the message, and includes the body of the original message in the reply (i.e., does not modify the contents). The subject field of the reply will already include <ISK> indicating that the reply is an Internet Sidekick scheduling message. In the body of the message, the user finds a section labeled "Your Reply." Here, the user simply types an "X" in the brackets next to Accept or Decline, so that the reply reads "[X] Accept" or "[X] Decline"; the replying user can also type a brief message in the space between the markers reading <BEGIN REPLY MESSAGE> and <END REPLY MESSAGE>. Now the user simply sends the reply. The originator's SK client will be able to process the reply automatically and mark the appropriate status for that recipient.

#### 5. Incoming Replies

The originator's SK client automatically collects reply messages from people which were invited to an event and displays the replies in the Activities view. As replies arrive, the user is notified by an icon in the News column (623 in FIG. 6A). The user can click the participants' names in the Activities view, and read their reply status, as well as any reply message they may have sent, in the Details and Information dialog 700, shown in FIG. 7A. By clicking the + sign by the participants in Details pane 701, the user can expand the list. By clicking a participant, the user can view the reply message in Information pane 703 and expand the Details and Information panes to full-screen, as desired.

The originator user can reschedule an event which he or she originated by selecting the event and clicking Reschedule button 711. This action displays a "Schedule an Event" dialog box where the user enters the new date and time information, and any other changes. The user is asked if he or she wants to notify all participants and resource managers about the change. If the user selects "Yes," notification is automatically sent.

In a similar manner, the user can cancel an event he or she originated by selecting the event and clicking Cancel Event button 713. The user will be asked to confirm that he or she wants to cancel the event and notify all participants; selecting "Yes" cancels the event. Alternatively, the user can delete group scheduling events in the Activities view to clean up his or her events list. This is done by selecting one or more events and pressing the Del key, or by right-clicking an event and choosing Delete from the menu. In contrast to canceling, the action of deleting group events in the Activities view does not remove them from the user's calendar, and does not notify meeting participants. It is used only to clean up the Activities list display. To delete a group event the user has originated and notify participants, the user clicks the Cancel Event button 713 or deletes the event in his or Calendar.

Also in the Details and Information dialog 700, the user can elect to send a reminder to all event participants in advance of a meeting. This is done by selecting the event and clicking Send Reminder button 715. Alternatively, the user can arrange for the reminder when the user initially creates the event.

If the recipient attaches a Free Time report (e.g., when declining or requesting rescheduling of the invitation), his or her Free Time report appears with the reply message, as indicated by Free Time button 721 in FIG. 7B. The Free Time report shows when the recipient has events booked and what times are free, during the next 30 days, as described above. It can be viewed by clicking on the Free Time button 721, whereupon the system displays Free Time report 723, for the reply. The report is useful in determining an alternate meeting time, especially when scheduling several people.

#### 6. Entering Replies Manually

For people who are invited by fax or phone, the user can enter the reply status manually. To enter reply information manually, the user selects the event from the list (in the Activities view). Next, the user selects the participant's name to bring up that individual's details in the Details pane. Now, the user can enter the status by right-clicking on the participant's name, to bring up pop-up menu 740, shown in FIG. 7C. Here, the user can enter: "Accepted," "Declined," "Reschedule Requested," "Delegated," or "No Reply Yet."

#### C. Resource Scheduling Interface

##### 1. General

Resources are facilities such as conference rooms or equipment that the user wants to be able to schedule. The system of the present invention groups resources in one of the following resource types:

1. Facilities—a resource such as a conference room or other immovable location. Facilities can have a Maximum Occupancy setting. The system maintains a calendar for facilities and confirms or denies bookings automatically based on the calendar.
2. Equipment—a movable resource such as a projector or vehicle. The system maintains a calendar for equipment and confirms or denies bookings automatically based on the calendar.
3. Other—private resource which an individual user wants to keep track of, such as booking a conference call operator, or ordering catering for a meeting. There is no calendar for this category, and entries are recorded under the user's To Do items.

Since resources typically do not have their own e-mail accounts, each resource has a "manager," whose e-mail is used to handle the calendar and messaging for reserving the resource. Typically, the manager is the person who creates the resource in the system of the present invention. Using the e-mail account of the manager, the system handles resource scheduling automatically in the background; the manager is not involved in these transactions. Alternatively, a resource can be given a dedicated e-mail account, whereupon the system uses it for automated scheduling of that resource. Once a resource has been added to the e-mail system (either to its own e-mail account or that of another's), it is distributed to others to make it available to them for booking.

##### 2. Reserving a Resource

The user can request a resource reservation as part of creating a group event. However, the user may want to do it separately, in advance, so he or she can be sure the resource is available before sending meeting invitations. To reserve a resource without scheduling the event, the user employs a Reservation Wizard of the present invention, which will now be illustrated.

To reserve a resource such as a conference room or projector using the wizard, the user selects Internet|Reservation Wizard from the system menu. This displays the first Resource Reservation Wizard dialog 800, Step 1 (Facilities and Equipment), as shown in FIG. 8A. The dialog lists resources which have already been distributed to

the user (and others), or have been created by the user. Here, the user chooses the facilities and equipment he or she wants to reserve by selecting each item. The user can reserve multiple resources by clicking each one. By clicking Details button 801, the user can see detailed information about a selected resource. When finished selecting resource, the user clicks "Next" button 803, to proceed to the next wizard dialog.

The second Resource Reservation Wizard dialog 810, Step 2, is illustrated in FIG. 8B. In Step 2 (Purpose and Time), the user enters the time and purpose for reserving the resource. When finished with Step 2, the user clicks Finish button 815. Based on the user's inputs, the Reservation Wizard sends out a request to book the resource for the time specified. The request is sent to the computer of the resource's manager (via e-mail account) for processing by the SK client. A reply message is sent back to the requestor's computer automatically, without requiring any action by the resource's manager. If the resource is available, the reply message confirms the reservation. If the resource is unavailable, a message listing available times is sent instead. In this case, the user can reschedule by again choosing Internet|Resources|Reservation Wizard and changing the times as needed before re-sending the request.

##### 3. Adding and Distributing Resources

The person who is going to manage the resource is usually the one who adds it using an Add command. This is selected from the system menu by choosing Internet|Resources|Add. In response, the system displays Add Resources dialog 820, illustrated in FIG. 8C. Here, the user enters the name of the resource together with the e-mail type and account information; the latter two are automatically picked up from existing user preferences. The phone number field and other fields are included for convenience in case others have questions about resource reservations. The process is completed by selecting Add button 821.

After a resource has been added, it can be distributed. "Distributing" a resource is the process of making it available to others for reserving. The manager distributes it to the users who will be reserving it. Anyone to whom the resource is distributed can distribute it to additional users. To distribute a resource to other users, the user selects a Distribute command from the menu. In response, the system displays Distribute Resources dialog 830, illustrated in FIG. 8D. At 831, the user clicks one or more resources that he or she manages and wants to distribute. At 833, the user selects users to distribute resources to.

##### 4. Managing Resources

The system of the present invention provides tools that let the user print resource schedules and modify or delete resources from his or her system. The user can view and print the schedule of any resource that he or she manages. To view a resource's schedule, the user chooses Internet|Resources|View. In response, the system displays Viewing Resources dialog 840, illustrated in FIG. 8E. From pulldown list 841, the user can select a particular resource to view and then click to print the schedule for the resource in weekly or monthly view.

To modify a resource (i.e., change the properties of a resource), the user chooses Internet|Resources|Modify from the menu. This opens a Modify Resource dialog box, which is similar to the Add Resource dialog box except that the user cannot modify the resource name. To remove a resource, the user chooses Internet|Resources|Remove, selects a resource from a displayed list, and clicks "Remove."

If the user modifies or removes a resource that he or she does not manage, the user is affecting the local resource file

only. The user might want to delete a resource that he or she never uses, or might want to change a resource's description or owner information as displayed on his or her system. Since these changes affect the user's local system only, they do not affect others who also reserve that resource. If the user is the owner of the resource, deleting it has wider impact. By deleting that resource, the user removes the resource's calendar, making it impossible for anyone to schedule that resource. If the user simply wants to stop managing the resource, he or she should "transfer" the resource.

Transferring a resource is the act of assigning a resource the user owns to another manager. This is done by choosing Internet\Resources\Transfer from the menu. In response, the system displays a Transfer Resources dialog 850, illustrated in FIG. 8F. Here, the user selects the resource from resource list 851 and then clicks Transfer button 853. The system now displays a second Transfer Resources dialog 860, shown in FIG. 8G, for entering information about the new resource manager. Transfer is completed by clicking Transfer button 861.

#### Internal Operation

##### A. Overview of Architecture and Basic Operation

FIG. 9 is a block diagram providing an overview of the internal architecture of a group scheduling system 900 constructed in accordance with the present invention. The group scheduling system 900 includes a main module 910 comprising a user interface 911, a composer 912, a parser/interpreter 913, an e-mail send/receive interface 914, a calendar interface 915, and an Activities view module 916. The user interface module 911 supports entering, editing, and deleting of group scheduling information. The composer 912 is employed for actually composing a group scheduling message. The message itself, when it is received by another SK client, is parsed and interpreted by the parser/interpreter 913. The e-mail module 914 supports the sending and receiving of e-mail messages and documents, using commercial available e-mail providers. The calendar interface 915 allows the main module or engine 910 to communicate with a separate calendar module 920. The Activities view module 916 supports the previously-demonstrated user sessions in Activities View mode of operation.

General operation of the system can be divided into two tasks: sending an event and receiving an event message. The general process of sending an event is illustrated by Send Event Method 1000, shown in FIG. 10. The process begins with the user entering event information in the user interface, as indicated by step 1001. The composer module, acting on the user information, proceeds to compose the event scheduling message, at step 1002. At the point of composing a message, the system provides a variety of different message types, each corresponding to a particular state a meeting is at. Once a message has been composed, it can be inserted into the calendar, as shown by step 1003. Thereafter, the system schedules the message to be sent via the e-mail transport layer, as indicated by "e-mail delivery" step 1004. Once the message has been sent, the Send Event Method 1000 is done.

The general process of receiving messages is illustrated by Receive Event Method 1100, illustrated in FIG. 11. At the outset, the process spawns a separate thread to receive an event message, at step 1101. As shown by step 1102, at a predetermined interval a "flash" session is initiated. During the flash session, the system polls the user's in-box, for identifying incoming scheduling messages (i.e., those having the signature <ISK>).

The incoming scheduling messages are parsed by the parser (913), as shown at step 1103. The parser extracts scheduling information from the messages, storing it in an internal representation suitable for use by other modules of

the system. The parser identifies, for instance, the <ISK> identifier, date and time information, message-type information (e.g., Schedule, Reschedule, Cancel, and so forth), and the like. The parsed information is, in turn, passed to a general message handler at step 1104. In response, the message handler triggers one of several actions, as indicated by step 1105, by invoking more-specific handlers—ones which perform the requested action. Actions include, for example, Schedule New 1105a, Reschedule 1105 B, Cancel 1105c, Minutes of Meeting 1105d, Resource Reservation 1105e, and the like. At step 1106, the parsed information is stored in the group scheduling database (950). From here, the information will also be employed, at step 1107, to update the calendar and/or a resource manager, as needed. Once updating is complete, the method is done.

##### B. Core Data Structures

##### 1. Group Item and Group Attachment

Before describing the internal methods of operation in further detail, it is first helpful to examine internal data structures which support operation of those methods. A group scheduling item is represented internally in the system by a "Group Item" data structure, GROUP\_ITEM. In an exemplary embodiment, this structure may be constructed as follows (using the C/C++ programming language, for instance).

```
typedef struct
{
    DWORD    dwEventID1;
    DWORD    dwEventID2;
    DWORD    dwFromDateTime;
    DWORD    dwToDateTime;
    DWORD    dwOldFromDateTime;
    DWORD    dwOldToDateTime;
    DWORD    dwSendReceive;
    HANDLE    hText;
    HGLOBAL    hRecipient;
    DWORD    lfData;
    LPGROUP_ATTACHMENT lpAttach;
    // READ_IN_MAIN; READ_IN_ATTACH; FTMPFILE;
    DWORD    wFlags;
    DWORD    wFlags2;
    short    nTexts;
    short    nRecipients;
    short    nPageHour;
    short    nPageMin;
    BYTE    bTypes;
    BYTE    bTimeZone;
    BYTE    bLead;
    BYTE    bLeadUnit;
} GROUP_ITEM; // total 64 bytes
typedef GROUP_ITEM * LPGROUP_ITEM;
```

As shown, the GROUP\_ITEM data structure includes two event IDs: dwEventID1 and dwEventID2. Together, these comprise a 64-bit (i.e., two double words) identifier for each scheduling group item. This ID is employed among the various clients for uniquely identifying a particular group scheduling item. Next, the data structure stores "from" and "to" date and time information in dwFromDateTime and dwToDateTime data members, respectively. Additionally, the particular item might comprise a "reschedule" item and, therefore, need to store the original ("old") date and time. This information is stored in dwOldFromDateTime and dwOldToDateTime data members. The dwSendReceive data member also stores date and time information; it is employed as a time stamp indicating when a group item is sent or received.

Following the date/time data members are two handles: hText and hRecipient. The hText data member is a handle to a text buffer for the subject line (e.g., 256-character string). The hRecipient data member, on the other hand, is a handle

pointing to a recipient block—a memory block storing N number of recipients. The lfdData member serves as a pointer to the detailed data—that is, the data which follows the subject line.

This is followed by another pointer, lpAttach, which 5 points to a group attachment data structure. The group attachment information is, therefore, nested within (via a pointer) each GROUP\_ITEM record. In an exemplary embodiment, a “Group Attachment” data structure, GROUP\_ATTACHMENT, may be constructed as follows. 10

```
#define ARRAY_LEN 63
typedef struct
{
    WORD wFlags; // IS_MEETINGNOTE_FILE
                  // if on then meeting note saved in file
                  // REMINDER_SENT
    short nAttachSize;
    DWORD wReserved;
    LPSTR lpAttachFile;
    HGLOBAL hRegarding;
    UINT nRegarding;
    LPSTR lpMeetingNote;
    UINT nMeetingNotes;
    char szLocation[ARRAY_LEN+1];
    char szCity[ARRAY_LEN+1];
} GROUP_ATTACHMENT; // 156 bytes;
typedef GROUP_ATTACHMENT * LPGROUP_ATTACHMENT;
```

As shown, the data structure includes as its first member nAttachSize, an integer storing a size for the attached file.

This is followed by wReserve which comprises a double word (DWORD) currently not used. The lpAttachFile data member stores a pointer to a text string comprising the name of the attach file (which itself is stored on disk).

The next data member, hRegarding, is a handle pointing to a “regarding” structure which comprises the message body—that is, what the event is in “regards” to. This is followed by an integer data member, nRegarding, which stores a size for the regarding block which is being pointed to by the hRegarding handle. The lpMeetingNote data member is a pointer to a text block comprising a user-supplied meeting note text string. The size of this pointed-to data structure is stored by nMeetingNotes. Finally, the location (e.g., particular conference room) and city where the meeting or event is to occur are stored by szLocation and szCity character strings, respectively.

Returning to description of GROUP\_ITEM data member, the next data member in the GROUP\_ITEM data structure is a double word, wFlags, storing a set of flags. The flags indicate various information for the group item and group appointment (described below), including, for example, whether the event is Canceled, Rescheduled, or Deleted. In an exemplary embodiment, the following flags are defined.

```
// flags for GROUP_APPOINTMENT and GROUP_ITEM.wFlags
#define READ_IN_MAIN 0x00000001 // both
#define READ_IN_ATTACH 0x00000002 // both
// #define FTMPFILE 0x0004 // both
#define HAS_MEETING_NOTE 0x00000008 // both
#define EVENT_CANCELED 0x00000010 // both
#define EVENT_RESCHEDULED 0x00000020 // both dwOldDateTime good
// #define EVENT_DELETED 0x00000040 // both
#define SCHEDULED_EVENT 0x00000080 // scheduler side
#define SEND_REMINDER 0x00000100 // scheduler side
#define REMINDER_SENT 0x00000200 // scheduler side
#define SEND_LATER 0x00000400 // scheduler side
#define NO_REPLY_NEEDED 0x00000800 // scheduler side
#define RECEIVED_EVENT 0x00001000 // receiver side
#define EVENT_ACCEPTED 0x00002000 // receiver side
#define EVENT_DECLINED 0x00004000 // receiver side
#define UNREAD_EVENT 0x00008000 // receiver side
#define EVENT_FORWARDED 0x00010000 // receiver side
#define SET_ALARM 0x00020000 // for GROUP_APPOINTMENT
only
#define STAMP_TO_CARD 0x00040000 // for GROUP_APPOINTMENT
only
#define ATTACH_MY_CALENDAR 0x00080000 // for GROUP_APPOINTMENT
only
#define RESOURCE_EVENT 0x00100000 // for GROUP_ITEM
#define FREETIME_EVENT 0x00200000
#define PERSONAL_EVENT 0x00800000
#define CONFIDENTIAL_EVENT 0x01000000
#define PAGE_ME 0x02000000
#define UNCONFIRMED_EVENT 0x04000000
#define COMPLETED_EVENT 0x08000000
#define UNBOOKED_EVENT 0x10000000
#define EVENT_RESCHEDULE_REQUIRED 0x20000000 // receiver side
// #define 0x40000000
#define EVENT_RECEIVE_FORWARD 0x80000000 // the second recipient
```

The flags can be combined (i.e., bitwise “or” operation), for 65 storing in a single double word.

The GROUP\_ITEM data structure stores a second set of flags in another double word data member, wFlags2. In an



exemplary embodiment, these additional flags are defined as follows.

```
// additional flags
#define EVENT_QUEUE_ITEM          0x00000001
#define EVENT_NEW_ITEM            0x00000002
#define EVENT_RECEIVE_RESCHEDULE  0x00000004
#define EVENT_RECEIVE_CANCEL      0x00000008
#define EVENT_RECEIVE_REMINDER    0x00000010 // receiver side
#define UNREAD_MEETING_NOTE       0x00000020
#define EVENT_RECEIVE_REPLY       0x00000040
```

Following the flags, the nText data member stores a byte count corresponding to the size of the block pointed to by the hText handle (described above). The nRecipients data member stores a count for the number of recipients; in an exemplary embodiment, up to 32,000 recipients are supported. The next two data members, nPageHour and nPageMin, specify a time to page the user; this corresponds to the “page me” reminder feature previously described (for the Scheduling Wizard, in FIG. 5H). In an exemplary embodiment, the page time is specified as a time interval in advance of the event or meeting, such as “three hours before” the meeting. This serves to remind the user so that he or she will not forget a meeting which he or she scheduled.

The next data member, bTypes, stores a value describing the item. In an exemplary embodiment, the following types are defined.

```
// values for bTypes in GROUP_ITEM and GROUP_APPOINTMENT
#define SCHEDULE_ITEM             1
#define BROADCAST_ITEM           2
#define RESCHEDULE_ITEM          3
#define REMINDER_ITEM            4
#define FREE_TIME_ITEM           5
#define CANCEL_ITEM              7
#define REPLY_ITEM               8
#define MINUTES_ITEM             9
#define EVENT_ACCEPT_ITEM        10
#define EVENT_DECLINE_ITEM       11
#define EVENT_FORWARD_REPLY_ITEM 12
#define EVENT_FORWARD_SCHEDULE_ITEM 13
#define EVENT_RESCHEDULE_REQUEST_ITEM 14
#define TRANSFER_SEND_ITEM       15
#define TRANSFER_ACCEPT_ITEM     16
#define TRANSFER_DECLINE_ITEM    17
#define DISTRIBUTE_ITEM          18
#define BOOK_SCHEDULE_ITEM       19
#define BOOK_RESCHEDULE_ITEM     20
#define BOOK_CANCEL_ITEM         21
#define BOOK_ACCEPT_ITEM         22
#define BOOK_DECLINE_ITEM        23
```

As shown, the bTypes data member can identify the item as a “Schedule” item, a “Broadcast” item, a “Reschedule” item, a “Reminder” item, a “Reply” item, a “Minutes” item, or the like.

The bTimeZone data member stores an identifier uniquely identifying the time zone (e.g., Pacific Standard Time) that the dates are relative to. In this manner, the recipient user’s system can automatically convert the times, if desired, to those relative to the time zone for that recipient. In a preferred environment, each appointment has its own time zone—typically, the time zone in which the item was originally scheduled. The recipient systems convert this information to and from their own relative time zones, as needed.

Finally, the bLead and bLeadUnit data members store information describing a “lead” time or reminder. This information allows the system to send reminders to all participants at a specified interval (e.g., three days).

## 2. Each Recipient

Each recipient involved with a group scheduling message is represented internally by a recipient data structure: EACH\_RECIPIENT. In an exemplary embodiment, the data structure may be constructed as follows.

```
typedef struct
{
    char    szName[32];
    char    szEmail[64];
    long    IID;
    long    IID2;
    DWORD   wFlags; // type of e-mail, Has free time,
                  // sender or not, file flag
    DWORD   wStatus; // status flags, message or free time saved.
    long    lIfData; // point to free time file and ShortMsg mail
    LPRECIDENT_DATA lpAttach;
    // HGLOBAL hAttach; // a RECIPIENT_DATA structure
} EACH_RECIPIENT; // 114
typedef EACH_RECIPIENT * LPEACH_RECIPIENT;
```

As shown, the first data member, szName, stores a text string comprising the recipient’s name (e.g., “John Smith”). This is followed by another text string, szEmail, which stores the e-mail address for the recipient. Following the recipient name and e-mail address are two identifier data members: IID and IID2. These IDs are used in combination to uniquely describe a particular recipient.

Following the identifiers is a double word data member, wFlags, storing recipient flags. The recipient flags indicate, for instance, whether the recipient has free time, whether the recipient accepts or declines, and so forth. In an exemplary embodiment, the status flags may be defined as follows.

```
// type of e-mails, fax, or page in EACH_RECIPIENT structure
#define TYPE_E-mail_AOL          0x0001
#define TYPE_E-mail_COMPUSERVE  0x0002
#define TYPE_E-mail_EXCHANGE     0x0004
#define TYPE_E-mail_INTERNET     0x0008
#define TYPE_FAX                 0x0010
#define TYPE_PAGE                 0x0020
#define TYPE_PHONE               0x0040
#define TYPE_MAILING_LIST        0x0080
// type of recipient
#define TYPE_PARTICIPANT         0x0100
#define TYPE_OC                  0x0200
#define TYPE_ROOM                0x0400
#define TYPE_EQUIPMENT           0x0800
// also, TYPE_OTHERS defined below
// misc. flags
```



-continued

```

#define IS_SENDER 0x1000
#define IS_SELF 0x2000
#define HAS_RECIPIENT_DATA 0x4000
#define READ_IN_RECIPIENT_DATA 0x8000
#define F_LIST_DIRTY 0x00010000
#define TYPE_OTHERS 0x00020000
#define CARD_ID_SET 0x00040000
#define TYPE_PAGE_ME 0x00080000
#define RECIPIENT_RECEIVED_FORWARD 0x00100000
#define RECIPIENT_SENT_FORWARD 0x00200000
// Recipient status flags
// #define RECIPIENT_STATUS 0x0001
#define RECIPIENT_FREE_TIME 0x0002
#define RECIPIENT_FORWARD 0x0004
#define RECIPIENT_E-mail_ADDRESS 0x0008
#define RECIPIENT_NAME 0x0010
#define RECIPIENT_CARD_ID 0x0020
#define RECIPIENT_MAIL_STATUS 0x0040
#define RECIPIENT_ACCEPT 0x0080
#define RECIPIENT_DECLINE 0x0100
#define RECIPIENT_RESCHEDULE_REQUIRED 0x0200
#define RECIPIENT_RESOURCE_CONFLICT 0x0400
#define RECIPIENT_RESOURCE_TRANSFERRED 0x0800
#define RECIPIENT_RESOURCE_NOT_FOUND 0x1000
#define RECIPIENT_RESOURCE_CANCELED 0x2000
// flags for resource events (from 0x00010000 to 0x80000000)
#define RECEIVE_DISTRIBUTE_RESOURCE 0x00010000
#define RECEIVE_BOOK_RESOURCE 0x00020000
#define SENT_BOOK_RESOURCE 0x00040000
#define RECEIVE_TRANSFER_RESOURCE 0x00080000
#define SENT_TRANSFER_RESOURCE 0x00100000
#define TRANSFER_SUCCESSFUL 0x00200000

```

As shown, the information indicates the type of delivery—e-mails, fax, or page—desired for a particular recipient, thereby supporting methods for communicating with all types of recipients, other than just e-mail. For e-mail types, support is provided for multiple providers, including, for instance, America On-line (AOL), CompuServe, Microsoft Exchange, and the Internet. For a recipient to be contacted via phone (i.e., type is TYPE\_PHONE), the system adds an item to the user's "call list"—that is, a list of calls the user is to make (as shown at 440, in FIG. 4). A delivery type of "mailing list" indicates that the system is to process multiple recipients according to a mailing list.

The flags also indicate a recipient type. A recipient can be, for instance, a "participant" (i.e., type is TYPE\_PARTICIPANT)—someone who is required to reply. A "CC" recipient (i.e., type is TYPE\_CC), on the other hand, is not required to reply. Other types of "recipients" include "room," such as a conference room, and "equipment," such as a projector. Finally, a recipient can be simply set to "others" for indicating that the recipient is other than that above.

As a group scheduling item is associated with a list of recipients, it is helpful to further characterize recipients. In this regard, a "sender" flag (type is IS\_SENDER) is employed for indicating that the recipient is actually the sender. A "self" flag (type is IS\_SELF), on the other hand, indicates that the recipient is "self"—that is, the same individual as the user. Other recipient-type information serves to further characterize the recipient. For instance, the "page me" flag indicates that this recipient is to be paged. A "sent forward" flag, on the other hand, indicates that this recipient has delegated the item to another user.

Finally, the recipient flags store status information for each recipient. Status includes, for instance, information indicating whether the recipient accepted or declined the

invitation, or whether the recipient required rescheduling. Additionally, at this point, flag information is stored for resources, such as information indicating that the resource is not found, that the resource has been transferred, that the resource has been canceled, or that a conflict for the resource exists.

Returning to the description of the EACH\_RECIPIENT data structure, the next data member, lIdData, is a pointer to a data block for the recipient; the data block stores free time information and a short message for the recipient. Finally, the EACH\_RECIPIENT data structure stores a pointer, lpAttach, which points to a RECIPIENT\_DATA structure. This structure will now be described.

### 3. Recipient Data

The RECIPIENT\_DATA structure stores free time and rescheduling information for a particular recipient. In an exemplary embodiment, the data structure may be defined as follows.

```

typedef struct
{
    DWORD dwFlags; // Reserved -- not used
    long lStart;
    long lEnd;
    DWORD dwRescheduleStart[3]; // reschedule request start
                                // date/time
    DWORD dwRescheduleEnd[3]; // reschedule request end
                                // date/time

    LPSTR lpMsg;
    LPSTR lpFreeTime;
    void * lpNext;
} RECIPIENT_DATA;
typedef RECIPIENT_DATA * LPRECIPIENT_DATA;

```

As shown, the first data member, dwFlags, comprises a double word data member storing flag or status information; it is currently not used. The next two data members, lStart and lEnd, indicate a free time range (e.g., next 30 days). Then, the next two data members, dwRescheduleStart and dwRescheduleEnd, store information specifying a reschedule request start date/time and reschedule request end date/time, respectively. As each of these data members comprises an array of size three, these data members actually indicate three rescheduling alternatives.

The next data member, lpMsg, comprises a pointer to a text string storing a message associated with the reschedule request (e.g., "Sorry, I am in Chicago next week; please reschedule."). This is followed by a pointer, lpFreeTime, which points to a block describing the free time for the recipient (i.e., information employed for displaying a Free Time report for this recipient). Finally, the data structure includes a "next" pointer, for pointing to a subsequent RECIPIENT\_DATA structure.

Each RECIPIENT\_DATA structure, when saved to file, is associated with a header: RECIPIENT\_DATA\_HEADER. In an exemplary embodiment, the header may be constructed as follows.

```

typedef struct
{
    DWORD dwFlags;
    UINT nMsgs;
    UINT nFreeTimes;
}

```

-continued

```

long    lStart;
long    lEnd;
DWORD   dwRescheduleStart[3]; // reschedule request
                                     start date/time
                                     5
DWORD   dwRescheduleEnd[3]; // reschedule request
                                     end date/time
} RECIPIENT_DATA_HEADER;
typedef RECIPIENT_DATA_HEADER *
LPRECIPIENT_DATA_HEADER;

```

The header includes data members the same as or similar to those described for RECIPIENT\_DATA. Additionally, the header stores a number of messages, nMsgs, and a number of free times, nFreeTimes.

Also stored is a footer: RECIPIENT\_DATA\_FOOTER. This may be constructed as follows.

```

typedef struct
{
    DWORD lfData; // -1 if no next item.
} RECIPIENT_DATA_FOOTER;
typedef RECIPIENT_DATA_HEADER *
LPRECIPIENT_DATA_HEADER;

```

The footer simply comprises a single data member, lfData, which serves to indicate whether a "next" recipient data item exists (in a chain of recipient data records).

#### 4. Group Appointment

At run time, a memory block or buffer is allocated for storing context information. This buffer is organized into a GROUP\_APPOINTMENT structure, for describing a particular group appointment. In an exemplary embodiment, this data structure may be constructed as follows.

```

typedef struct
{
    DWORD   dwEventID1;
    DWORD   dwEventID2;
    DWORD   dwFromDateTime;
    DWORD   dwToDateTime;
    DWORD   dwOldFromDateTime;
    DWORD   dwOldToDateTime;
    DWORD   dwRescheduleStart[3];
    DWORD   dwRescheduleEnd[3];
    DWORD   dwSendReceive;
    HANDLE  hText;
    HANDLE  hRegarding;
    HANDLE  hRecipient;
    LPSTR   lpMeetingNote; // meeting note file name
    LPSTR   lpShortMsg;
    LPSTR   lpDelegateMsg;
    LPSTR   lpFreeTime;
    int     nIsDelegate;
    int     nIsResource;
    HANDLE  hForward;
    WORD    nTexts;
    WORD    nRegardings;
    WORD    nRecipients;
    DWORD   wFlags; // READ_IN
    DWORD   wFlags2;
}

```

-continued

```

short     nPageHour;
short     nPageMin;
BYTE      bLeadHour;
BYTE      bLeadMin;
BYTE      bReminderLead;
BYTE      bReminderUnit;
BYTE      bTimeZone;
BYTE      bTypes;
int       nTone;
DWORD     lfData; // 42
int       nResourceStatus;
char      szResourceName[64];
char      szLocation[64];
char      szCity[64];
char      szSenderName[32];
char      szSenderE-mail[64];
int       nSenderE-mailType;
char      szAttachFile[PATHMAX+4];
EACH_RECIPIENT PageMe;
HGLOBAL  hPlaceList;
int       cPlaceLists;
// LPSTR lpTZ;
// int nTZs;
} GROUP_APPOINTMENT; // total 136 bytes
typedef GROUP_APPOINTMENT * LPGROUP_APPOINTMENT;

```

This temporary, in-memory data structure (i.e., it is not saved to disk) largely comprises data members taken from the previously-described data structures. Additionally, however, the data structure stores information further describing any delegation. For instance, the lpDelegateMsg data member points to a delegate message text string. This allows one recipient to send a message to a delegated-to recipient, yet not have that text string message returned to the originator of the invitation. Additionally, delegation is supported by the szSenderName and szSenderE-mail data members. This information indicates who the delegated to recipient should respond to—that is, the originator of the invitation.

#### Group Scheduling Methodology

##### 1. Schedule Group Event

With an understanding of the internal day structures employed, methods of operation may now be examined in further detail. Group scheduling begins with a request to schedule a group event. This request results from user input occurring at the user interface, as illustrated at step 1001 in FIG. 10. Actual operation of the user interface itself, such as processing menu selections and user input at edit fields, may be done in a conventional manner using standard windows graphical user interface (GUI) methodology.

Once the user input has culminated in a request to schedule a group event, the request is processed by the group scheduling module or engine 910 (of FIG. 9). This module internally invokes a Schedule\_Group\_Event method. In an exemplary embodiment, the Schedule\_Group\_Event handler or method may be constructed as follows (again, using the C/C++ programming language).

```

1: /*****
2:  Schedule a following type of group events:
3:  - Normal group event (reply required)
4:  - Broadcast group event (no reply needed)
5:  - Reminder
6:  - Reschedule

```

```

7:   - Free time
8:   - Cancel meeting.
9:   .....
10:  int Schedule_Group_Event (LPGROUP_APPOINTMENT lpNew,
11:    int nAddToList, int nAddToAppointment, int nFromCalendar,
12:    LPSTR lpInBuf, int nInSize)
13:  {
14:    GROUP_ITEM GroupItem;
15:    LPSTR lpMessage = lpInBuf;
16:    int nIndex = -1;
17:
18:    if (!lpMessage)
19:    {
20:      lpMessage = AllocLockBufferPtr (NOTETEXTSIZE * 3);
21:      nInSize = NOTETEXTSIZE * 3;
22:    }
23:
24:    if (!lpMessage)
25:      return FALSE;
26:
27:    nStartGroupEvent = TRUE;
28:
29:    // Save to appointment list in Calendar module.
30:    if (lpNew->hRecipient && lpNew->nRecipients)
31:    {
32:      // get send and receive time.
33:      if (lpNew->bTypes == SCHEDULE_ITEM || lpNew->bTypes
34:        == BROADCAST_ITEM || lpNew->bTypes
35:        == RESCHEDULE_ITEM || lpNew->bTypes == CANCEL_ITEM)
36:        time (&(lpNew->dwSendReceive));
37:
38:      // - add call items to call list.
39:      if (lpNew->bTypes == SCHEDULE_ITEM
40:        || lpNew->bTypes == BROADCAST_ITEM)
41:        Insert_Calls_Calendar (lpNew);
42:
43:      // - add send letter to todo list.
44:      if (lpNew->bTypes == SCHEDULE_ITEM
45:        || lpNew->bTypes == BROADCAST_ITEM)
46:        Insert_Todos_Calendar (lpNew);
47:
48:      // add/edit/delete calendar item
49:      if (lpNew->nFromCalendar)
50:      {
51:        if ((lpNew->bTypes == SCHEDULE_ITEM
52:          || lpNew->bTypes == BROADCAST_ITEM)
53:          && nAddToAppointment)
54:        {
55:          if (!(lpNew->wFlags & UNBOOKED_EVENT))
56:            AddNewAppointment_FromGROUP (lpNew);
57:          // AddNewGroupAppointment (lpNew);
58:        }
59:        else if (lpNew->bTypes == RESCHEDULE_ITEM)
60:        {
61:          // #### Reschedule Appointment
62:          ChangeExistAppointment_FromGROUP (lpNew);
63:        }
64:        else if (lpNew->bTypes == CANCEL_ITEM)
65:        {
66:          // #### delete appointment
67:          DeleteExistAppointment_FromGROUP (lpNew);
68:        }
69:      }
70:
71:      // add item to GROUP event data base.
72:      if (nAddToList)
73:      {
74:        ResolveAllMailingLists (lpNew);
75:        if (!GROUP_Appointment_To_GROUP_Item (lpNew, &GroupItem))
76:        {
77:          UnlockFreeBufferPtr (lpMessage);
78:          return FALSE;
79:        }
80:        GroupItem.wFlags == READ_IN_MAIN | READ_IN_ATTACH;
81:        GroupItem.wFlags == SCHEDULED_EVENT;
82:        GroupItem.wFlags2 == EVENT_QUEUE_ITEM;
83:        if (!IsBroadcastType (&GroupItem))
84:        {
85:          ResolveAllExchangeMailingLists (&GroupItem);
86:        }

```

-continued

```

87:
88:     ScanForSelf_SetAccept (&GroupItem);
89:     // actual call to add item to GROUP event database
90:     nIndex
91:         = AddOneGroupItem ( NULL, &GroupItem,
92:                             SCHEDULED_EVENT,
93:                             Get_Current_SortFlag ( ));
94:     // ... DEBUG
95: }
96: // empty else case ...
97: // Now COMPOSE MESSAGE
98: if (!(lpNew->wFlags & SEND_LATER))
99: {
100:     if (lpNew->bTypes == SCHEDULE_ITEM)
101:     {
102:         ComposeScheduleMessage (lpNew,
103:                                 MSG_SCHEDULE, lpMessage, nInSize);
104:     }
105:     else if (lpNew->bTypes == BROADCAST_ITEM)
106:     {
107:         ComposeScheduleMessage (lpNew, MSG_BROADCAST,
108:                                 lpMessage, nInSize);
109:     }
110:     else if (lpNew->bTypes == REMINDER_ITEM)
111:     {
112:         ComposeScheduleMessage (lpNew, MSG_REMINDER,
113:                                 lpMessage, nInSize);
114:         bIsReminderMsg = TRUE;
115:     }
116:     else if (lpNew->bTypes == RESCHEDULE_ITEM)
117:     {
118:         ComposeScheduleMessage (lpNew, MSG_RESCHEDULE,
119:                                 lpMessage, nInSize);
120:     }
121:     else if (lpNew->bTypes == CANCEL_ITEM)
122:     {
123:         ComposeCancelMessage (lpNew, lpMessage, nInSize);
124:     }
125:
126:     // - send email;
127:     Send_AOL_Email (lpNew, lpMessage);
128:     Send_CompuServe_Email (lpNew, lpMessage, FALSE);
129:     Send_Exchange_Email (lpNew, lpMessage, FALSE);
130:     Send_Internet_Email (lpNew, lpMessage, FALSE);
131:
132:
133:     // - send fax;
134:     Send_Group_Fax (lpNew, lpMessage, FALSE);
135:
136:     if ((lpNew->bTypes == SCHEDULE_ITEM) ||
137:         (lpNew->bTypes == BROADCAST_ITEM) ||
138:         (lpNew->bTypes == RESCHEDULE_ITEM) ||
139:         (lpNew->bTypes == CANCEL_ITEM))
140:     {
141:         int nLen = lstrlen (lpMessage);
142:
143:         Loadstring (hCardfileInstance,
144:                     IDS_THIS_IS_RESOURCE, lpMessage + nLen, BUF_LEN);
145:
146:         Send_CompuServe_Email (lpNew, lpMessage, TRUE);
147:         Send_Exchange_Email (lpNew, lpMessage, TRUE);
148:         Send_Internet_Email (lpNew, lpMessage, TRUE);
149:         Send_Group_Fax (lpNew, lpMessage, TRUE);
150:     }
151:
152:     // - send page;
153:     Send_Group_Page (lpNew, FALSE);
154:     bIsReminderMsg = FALSE;
155: }
156: fScheduledDirty = TRUE;
157: }
158: else
159: {
160:     AddNewAppointment FromGROUP (lpNew);
161: }
162: if (!lpInBuf)
163:     UnlockFreeBufferPtr (lpMessage);

```

```

164:  nStartGroupEvent = FALSE;
165:
166:  return TRUE;
167: }

```

(line numbers added above to aid in the following description)

This method is invoked after the user has completed UI data entry. It serves to schedule one of the following types of group events: Normal group event (reply required), Broadcast group event (no reply required), Reminder, Reschedule, Free Time, or Cancel meeting.

The parameters required for the method are set forth at lines 10–12. The first parameter, *lpNew*, references a new group appointment data structure (described above). The second parameter, *nAddToList*, indicates that this group event should be added to the group scheduling database. The third parameter, *nAddToAppointment*, indicates whether the event should be added to the user's appointments (listed in the user's calendar). The fourth parameter, *nFromCalendar*, indicates that this request is originating from the user's calendar, for example, as a result of a drag-and-drop event occurring in the calendar. The first two parameters, *lpnBuff* and *nSize*, characterize a passed-in memory buffer; the memory buffer is typically used for storing message strings. At lines 14–16, the method initializes local (stack) variables. At lines 18–25, the method attempts to allocate a memory buffer for storing a message. If a buffer cannot be successfully allocated, the method returns "false" at line 25. At line 27, the method initializes a global variable, *nStartGroupEvent*, to the value of "true."

Starting at line 29, the method will undertake to save the group event to the appointment list in the calendar module. This occurs as follows. At line 30, the method tests whether there are recipients—that is, whether this is a "group" appointment. Otherwise (i.e., the condition at line 30 is (false)), the event is instead a "local" appointment. Between lines 32 and 69, the method tests group appointment flags for determining its course of action. At lines 32–36, for instance, if the event is a "Schedule" item, a "Broadcast" item, a "Reschedule" item, or a "Cancel" item, the previously-described *dwSendRecieve* time stamp is updated at line 36. At lines 38–41, the method adds a Schedule or Broadcast item to the user's call list. In a similar manner, at lines 43–46, the method adds a Schedule or Broadcast item to the user's "to do" list. The "insert calendar" call serves to call into the calendar module 920 (of FIG. 9).

At lines 48–49, the method tests whether the event request originated from the calendar. If not, then the method must add the event to the appointment list, if the requested event is a Schedule or Broadcast item. Specifically, the method invokes an *AddNewAppointment\_FromGROUP* subroutine at line 56. Otherwise, the method tests whether the event is a Reschedule item (at line 59) or a Cancel item (at line 64). For a Reschedule item, the method changes the existing appointment, at line 62. For a Cancel item, on the other hand, the method deletes the existing appointment, at line 67.

At lines 71–95, the method undertakes to add the event item to the group scheduling (event) database 950 (of FIG. 9). Specifically, the method tests the *nAddToList* parameter, at line 72. If this is set to "true" (i.e., greater than 0), the method proceeds to convert the group appointment into a group item, resolving all recipients on any mailing lists. The actual call to add the item to the database occurs at lines 89–93, where one group item is added.

Starting with line 97, the method will now undertake composing of the message for notifying recipients of the group scheduling event. At line 98, the method tests a *SEND—LATER* flag, which indicates whether the scheduling message is to be sent later. If the message is not to be sent later (i.e., the "if" statement at line 98 evaluates to "true"), then the group scheduling event message must be composed now. In such a case, the method will proceed to compose and send a message, at lines 99–155. Steps of the method for composing and sending the message will be described next.

Lines 100–124 set forth a sequence of "else if" case-like statements which switch on the item type. At line 100, for instance, if the item is a Schedule item, the method invokes the composer at lines 102–103, for composing a Schedule message (*MSG\_SCHEDULE*). If, instead, the item is a Broadcast item, tested at line 105, the method invokes the composer at lines 107–108, for composing a Broadcast message (*MSG—BROADCAST*). For a Reminder item, tested at line 110, the method invokes the composer at lines 112–113, for composing a reminder message (*MSG\_REMINDER*). For a Reschedule item, tested at line 116, the method invokes the composer at lines 118–119, for composing a reschedule message (*MSG\_RESCHEDULE*). Finally, if the item is a Cancel item, tested at line 121, the method invokes the composer at line 123, for composing a cancel message. The foregoing steps, therefore, invoke an appropriate handler in the composer 912 (of FIG. 9) for creating an appropriate message. The message itself is stored in the memory buffer pointed to by *lpMessage*—the memory buffer allocated at lines 18–25. After the task of composing the message is done, the method proceeds to send the message using the appropriate transport for each recipient.

At lines 126–130, the method invokes the e-mail module or manager for transporting the message over available services: America On-Line (handler invoked at line 127), CompuServe (handler invoked at line 128), Microsoft Exchange (handler invoked at line 129), and Internet (handler invoked at line 130). As shown, each handler invocation passes a pointer to the *GROUP\_\_APPOINTMENT* data structure as well as a pointer to the message buffer (which holds the just-created group scheduling event message). As shown, a Boolean parameter is passed with the value of "false" for indicating that the message is not for a resource (which requires a slightly different format). For each handler invocation, the e-mail module will enumerate the recipients for the respective services and post group scheduling event messages accordingly.

At lines 133–134, the method invokes a fax handler for sending facsimile transmissions to the recipients which are preferably reached via that mechanism. Faxing from the computer can be done in a conventional manner, using a "Fax modem" and built-in Fax support (e.g., in Microsoft Windows® 95, available from Microsoft Corp. of Redmond, Wash.).

At lines 136–150, the method essentially repeats the foregoing steps of invoking e-mail and fax handlers, except that here the process is undertaken for a resource. The process is essentially the same except that at lines 141–144

the method modifies the just-created message (stored in the memory buffer) for adding an identifier indicating that the group scheduling event is for a resource. At lines 152-153, the method invokes a SendGroupPage handler for processing any request to page a recipient (or the user himself or herself). At lines 154 and 156 local cleanup is performed.

At line 158, an "else" statement is defined. This statement is reached in the event that the message is to be sent later (i.e., the "if" statement of line 98 evaluates to "false"). In such a case, the method simply adds a new appointment, at line 160. Steps are not undertaken at this point to compose and/or post a group scheduling event message. Finally, the method performs local cleanup, such as freeing the message buffer, at lines 162-163. After resetting the nStartGroupE-

calendar with the embedded calendar. The synchronization generic in nature, so that it can be extended to any data object which is capable of synchronization.

#### 5 b. Message Body

Following the signature and message-type is (if required) a message body. Here, the system includes specific data items which characterize the scheduling event. The message body for Schedule and Broadcast message-types is exemplary in this regard. In an exemplary environment, a message body for these message-types may be constructed as follows:

---

<From Date Time>	:=	YY/MM/DD HH:MM
<To Date Time>	:=	YY/MM/DD HH:MM
<Location>	:=	Place name where the meeting will be held
<City>	:=	City name where the meeting will be held It is related to Time Zone.
<Time Zone>	:=	<Time zone name where appointment time is based on>
<Attributes>	:=	<Alarm>
<EventID1>	:=	A unique ID for the event.
<EventID2>	:=	The second part of the ID.
<Subject>	:=	<Message text></Subject>
<Message Content>	:=	<Regarding text></Message Content>
<Participants>	:=	"Name; Email address; Email Type" "Name; Email Address; Email Type" </Participants>
<CC>	:=	"Name; Email address; Email Type" "Name; Email Address; Email Type" </CC>
<Room>	:=	"Name; Email address; Email Type" "Name; Email Address; Email Type" </Room>
<Equipment>	:=	"Name; Email address; Email Type" "Name; Email Address; Email Type" </Equipment>

---

vent global flag, at line 164, the method returns "true," at line 166, whereupon the method is completed.

#### 2. Composing Messages

To understand methods of the present invention for composing messages, it is helpful at the outset to examine group scheduling formats employed by the system for creating messages. In general, messages are created using delimiters, thereby, avoiding position dependency of particular information within a message.

##### a. Message Formats

In an exemplary environment, every message includes a signature and a message type, which may be constructed as follows:

---

<Product Signature>	:=	<ISK> (or <SIS>)
<Message Type>	:=	<Schedule/Broadcast/Reply/Free Time/Reminder/Reschedule/Meeting Note/Cancel/Recurring/New Resource/Resource/Action/Synchronization>

---

As shown, the message includes the previously-described signature followed by a particular message-type. The message-type itself can be any one of the following: Schedule, Broadcast, Reply, Free Time, Reminder, Reschedule, Meeting Note, Cancel, Recurring, New Resource, Resource, Action, and Synchronization. The "Action" message-type supports workflow applications. Here, the message serves as a "to do" item or task for one or more recipient users. The "Synchronization" message, on the other hand, serves to synchronize two or more data objects, such as user calendars. Recall that a data object characterizing a calendar (e.g., free time data object) can be imbedded within a message. A Synchronization message-type, therefore, exchanges calendars and instructs the recipient system to automatically synchronize the recipient's

As shown, the message body includes data items which correspond to the previously-described internal data structures. For instance, the group scheduling event is characterized by a "from" and a "to" date and time, a location, a time zone, and the like. The event is identified by the previously-described two-part event ID. The message body concludes with a list of e-mail addresses, including addresses for participants, "CC" recipients, and resources (divided into "room" and "equipment" subtypes). Each of these data items may comprise a list of e-mail addresses.

The message body for a reply message, on the other hand, is constructed as follows.

---

<Reply Types>	:=	<Accept/Decline/Reschedule Required/Sleep/Forward/Resource/Free Time>
<Forward To>	:=	<Name; E-mail address; Email Type>
</Forward To>		
<< If Reschedule required		
<From Date Time>	:=	YY/MM/DD HH:MM
<To Date Time>	:=	YY/MM/DD HH:MM
>>		
<EventID1>	:=	A unique ID for the event.
<EventID2>	:=	The second part of the ID.
<Short Message>	:=	<Short text message></Short Message>

---

Again, the message body includes data items corresponding to the previously-described data members in the group scheduling internal data structures. A reply includes, for instance, a reply type, such as "Accept," "Decline," "Reschedule required," or the like. In the event that the replying user has delegated the event, the delegated-to user's address is provided in the "forward to" field. If the reply

request rescheduling, rescheduling date and time information is included. For this type of message, the message also includes a file attachment providing further information for (e.g., Free Time report). Finally, the message includes the two-part event ID, so that the reply can be matched up with the original invitation. Any message provided by the replying user is transmitted in the message field, "short message."

The "sleep" message-type is provided for those users who are on vacation. Exemplary message bodies for other message types are appended herewith as Appendix A.

c. Methodology for Composing Messages

In an exemplary environment, a method for composing scheduling event messages, ComposedScheduledMessage, may be constructed as follows.

```

1:  /*.....
2:  This function is called for schedule a:
3:  - Schedule meeting
4:  - Broadcast meeting
5:  - Reminder
6:  - Reschedule
7:
8:  Meeting message format is:
9:
10: <Message Type> message type name\r\n
11: <Old Date Time> yy/mm/dd hh:mm\r\n (if it's a reschedule message)
12: <From Date Time> yy/mm/dd hh:mm\r\n
13: <To Date Time> yy/mm/dd hh:mm\r\n
14: <Location> location name\r\n
15: <Time Zone> time zone value\r\n
16: <Attributes> "attribute" "attribute" ... \r\n
17: <EventID1> EventID1 value string\r\n
18: <EventID2> EventID2 value string\r\n
19: <Subject> Text body</Subject>\r\n
20: <Message Content> Regarding body</Message Content>\r\n
21:
22: ...../
23:
24: int ComposeScheduleMessage (LPGROUP_APPOINTMENT lpGroup,
25:                             int nMessageType,
26:                             LPSTR lpMessage,
27:                             int nSize)
28: {
29:     char    szField[BUF_LEN+1], szSpace[2],
30:             szReturn[5], szTime[BUF_LEN+1];
31:     int     nIsReschedule = FALSE, nIsBroadCast = FALSE;
32:     LPSTR   lpBuf;
33:     BYTE    bTypes;
34:
35:     szSpace[0] = ' ';
36:     szSpace[1] = 0;
37:     GetReturnNewline (szReturn);
38:
39:     if ((nMessageType == MSG_DISTRIBUTE_RESOURCE)
40:         || (nMessageType == MSG_TRANSFER_RESOURCE))
41:     {
42:         LoadGenericHeaderMessage (lpMessage);
43:     }
44:     else
45:     {
46:         if (nMessageType == MSG_SCHEDULE ||
47:             nMessageType == MSG_REMINDER ||
48:             nMessageType == MSG_BROADCAST ||
49:             nMessageType == MSG_RESCHEDULE)
50:             Load_ISK_Message_Header (lpGroup, lpMessage, nSize);
51:     }
52:
53:
54:     /*
55:
56:     This message was generated by Internet Sidekick.
57:     * If you have Internet Sidekick installed on your system:
58:     - Please ignore this message and do not delete it from you inbox.
59:     - Internet Sidekick will automatically process it next time you
60:       Send/Receive messages or during the next scheduled Flash Session.
61:
62:     * If you do not have Internet Sidekick:
63:     - You can reply to this invitation using your e-mail software.
64:       Type an X next to either Accept or Decline below, so it reads
65:       [X] Accept or [X] Decline.
66:     - You can also enter a short message as part of your reply to the
67:       initiator in the blank section between <BEGIN REPLY MESSAGE> and
68:       <END REPLY MESSAGE>.
69:     - When you reply from your E-mail product, make sure that the reply
70:       includes the whole original message body including the section

```

```

71: below the heading, For Internet Sidekick Use Only. Also do not
72: modify or delete the Subject of this e-mail message.
73: - For further information on how to procure it, please visit
74: Starfish Software's web site at:
75: <http://www.starfishsoftware.com>.
76: .....
77:
78: Message From: {Initiator}
79: Date/Time: {Date/Time}
80: Duration: {Duration}
81: Time Zone: {Time Zone}
82: Location: {Location}
83: City/Country: {City}
84:
85: Subject: {Subject}
86: Message: {Message}
87:
88: .....
89:
90: <BEGIN REPLY>
91: Your Reply:
92:
93: [ ] Accept
94: [ ] Decline
95: <END REPLY>
96:
97: Reply Message:
98: <BEGIN REPLY MESSAGE>
99:
100: <END REPLY MESSAGE>
101: .....
102: The section below is for Internet Sidekick Use Only.
103: Please, do not edit or delete any of the information
104: below this line.
105: .....
106:
107: */
108:
109:
110: // append <Message Type> and space
111: LoadString (hCardfileInstance,
112:             IDS_SKWGROUP_MESSAGE_TYPE_ID,
113:             lpMessage+strlen(lpMessage), nSize);
114: lstrcat (lpMessage, szSpace);
115:
116: // append Schedule, Broadcast, Reminder, Reschedule
117: switch (nMessageType)
118: {
119: case MSG_SCHEDULE:
120:     LoadString (hCardfileInstance,
121:                 IDS_SKWGROUP_SCHEDULE, szField, BUF_LEN);
122:     break;
123: case MSG_BROADCAST:
124:     LoadString (hCardfileInstance,
125:                 IDS_SKWGROUP_BROADCAST,
126:                 szField, BUF_LEN);
127:     nlsBroadCast = TRUE;
128:     break;
129: case MSG_REMINDER:
130:     LoadString (hCardfileInstance,
131:                 IDS_SKWGROUP_REMINDER, szField, BUF_LEN);
132:     break;
133: case MSG_RESCHEDULE:
134:     LoadString (hCardfileInstance,
135:                 IDS_SKWGROUP_RESCHEDULE, szField, BUF_LEN);
136:     nlsReschedule = TRUE;
137:     break;
138: case MSG_RESOURCE:
139:     LoadString (hCardfileInstance,
140:                 IDS_SKWGROUP_RESOURCE_SCHEDULE,
141:                 szField, BUF_LEN);
142:     break;
143: case MSG_DISTRIBUTE_RESOURCE:
144:     LoadString (hCardfileInstance,
145:                 IDS_SKWGROUP_NEW_RESOURCE,
146:                 szField, BUF_LEN);
147:     break;
148: case MSG_RESOURCE_RESCHEDULE:
149:     LoadString (hCardfileInstance,
150:                 IDS_SKWGROUP_RESOURCE_RESCHEDULE,

```



-continued

```

151:         szField, BUF_LEN);
152:     break;
153: case MSG_RESOURCE_CANCEL:
154:     LoadString (hCardfileInstance,
155:         IDS_SKWGROUP_RESOURCE_CANCEL,
156:         szField, BUF_LEN);
157:     break;
158: }
159: AppendString (lpMessage, szField,
160:     szReturn, NULL, NULL, NULL, nSize);
161:
162: if (nIsReschedule)
163: {
164:     /*
165:     if it is a reschedule operation
166:     then insert original date/time
167:     */
168:     GetGmtTimeString (lpGroup->dwOldFromDate,
169:         lpGroup->bTimeZone, szTime, BUF_LEN);
170:     LoadString (hCardfileInstance,
171:         IDS_SKWGROUP_OLD_DATE_TIME,
172:         szField, BUF_LEN);
173:     AppendString (lpMessage, szField,
174:         szSpace, szTime,
175:         szReturn, NULL, nSize);
176: }
177:
178: // Append from and to date/time
179: LoadString (hCardfileInstance,
180:     IDS_SKWGROUP_FROM_DATE_TIME,
181:     szField, BUF_LEN);
182: GetGmtTimeString (lpGroup->dwFromDate,
183:     lpGroup->bTimeZone, szTime, BUF_LEN);
184: AppendString (lpMessage, szField,
185:     szSpace, szTime, szReturn, NULL, nSize);
186:
187: LoadString (hCardfileInstance,
188:     IDS_SKWGROUP_TO_DATE_TIME, szField, BUF_LEN);
189: GetGmtTimeString (lpGroup->dwToDate,
190:     lpGroup->bTimeZone, szTime, BUF_LEN);
191: AppendString (lpMessage, szField,
192:     szSpace, szTime, szReturn, NULL, nSize);
193:
194: // Append city and country information
195: LoadString (hCardfileInstance,
196:     IDS_SKWGROUP_CITY, szField, BUF_LEN);
197: AppendString (lpMessage, szField,
198:     szSpace, lpGroup->szCity,
199:     szReturn, NULL, nSize);
200:
201: // Append location and time zone information
202: // <Location> location name\r\n
203: // <Time Zone> time zone value\r\n
204: // IDS_SKWGROUP_TIME_ZONE, "<Time Zone>"
205: // IDS_SKWGROUP_LOCATION, "<Location>"
206: LoadString (hCardfileInstance,
207:     IDS_SKWGROUP_LOCATION, szField, BUF_LEN);
208: AppendString (lpMessage, szField,
209:     szSpace, lpGroup->szLocation,
210:     szReturn, NULL, nSize);
211: LoadString (hCardfileInstance,
212:     IDS_SKWGROUP_TIME_ZONE, szField, BUF_LEN);
213: // nvalue = GetTimeZoneValue (lpGroup);
214: _itoa ((int)lpGroup->bTimeZone, szTime, 10);
215: AppendString (lpMessage, szField,
216:     szSpace, szTime, szReturn, NULL, nSize);
217:
218: // Append attributes
219: // <Attributes> "attribute" "attribute" ... \r\n
220: // IDS_SKWGROUP_ATTRIBUTES, "<Attributes>"
221: LoadString (hCardfileInstance,
222:     IDS_SKWGROUP_ATTRIBUTES, szField, BUF_LEN);
223: GetAttributeString (lpGroup, szTime, BUF_LEN);
224: AppendString (lpMessage, szField,
225:     szSpace, szTime, szReturn, NULL, nSize);
226:
227: // Append Event ID1
228: // <EventID1> EventID1 value string\r\n
229: // IDS_SKWGROUP_EVENTID_1, "<EventID1>"
230: LoadString (hCardfileInstance,

```

-continued

```

231:         IDS_SKWGROUP_EVENTID_1, szField, BUF_LEN);
232:     _ltoa ((long)lpGroup->dwEventID1, szTime, 10);
233:     AppendString (lpMessage, szField,
234:         szSpace, szTime, szReturn, NULL, nSize);
235:
236:     // Append Event ID2
237:     // <EventID2> EventID2 value string\r\n
238:     // IDS_SKWGROUP_EVENTID_2, "<EventID2>"
239:     LoadString (hCardfileInstance,
240:         IDS_SKWGROUP_EVENTID_2, szField, BUF_LEN);
241:     _ltoa ((long)lpGroup->dwEventID2, szTime, 10);
242:     AppendString (lpMessage, szField,
243:         szSpace, szTime,
244:         szReturn, NULL, nSize);
245:
246:     if (lpGroup->wFlags & EVENT_FORWARDED)
247:     {
248:         EACH_RECIPIENT Recipient;
249:
250:         // append sender's name and sender's email address
251:         if (GetSenderInfo (lpGroup, &Recipient, IS_SENDER) >= 0)
252:         {
253:             LoadString (hCardfileInstance,
254:                 IDS_SKWGROUP_SENDER_NAME,
255:                 szField, BUF_LEN);
256:             AppendString (lpMessage, szField,
257:                 szSpace, Recipient, szName,
258:                 szReturn, NULL, nSize);
259:             LoadString (hCardfileInstance,
260:                 IDS_SKWGROUP_SENDER_EMAIL,
261:                 szField, BUF_LEN);
262:             AppendString (lpMessage, szField,
263:                 szSpace, Recipient, szEmail,
264:                 szReturn, NULL, nSize);
265:         }
266:         // append forward flag "1"
267:         LoadString (hCardfileInstance,
268:             IDS_SKWGROUP_JS_DELEGATE,
269:             szField, BUF_LEN);
270:         AppendString (lpMessage, szField, szSpace,
271:             "1", szReturn, NULL, nSize);
272:     }
273:
274:     // Append Text
275:     // <Subject> Text body</Subject>\r\n
276:     // IDS_SKWGROUP_TEXT, "<Subject>"
277:     // IDS_SKWGROUP_END_TEXT, "</Subject>"
278:     if (lpGroup->hText)
279:     {
280:         LoadString (hCardfileInstance,
281:             IDS_SKWGROUP_TEXT,
282:             szField, BUF_LEN);
283:         LoadString (hCardfileInstance,
284:             IDS_SKWGROUP_END_TEXT,
285:             szTime, BUF_LEN);
286:         lpBuf = GlobalLock (lpGroup->hText);
287:         AppendString (lpMessage, szField,
288:             szSpace, lpBuf,
289:             szTime, szReturn, nSize);
290:         GlobalUnlock (lpGroup->hText);
291:     }
292:
293:     // Append Regarding
294:     // <Message Content> Regarding body</Message Content>\r\n
295:     // IDS_SKWGROUP_REGARDING, "<Message Content>"
296:     // IDS_SKWGROUP_END_REGARDING, "</Message Content>"
297:     if (lpGroup->hRegarding)
298:     {
299:         LoadString (hCardfileInstance,
300:             IDS_SKWGROUP_REGARDING, szField, BUF_LEN);
301:         LoadString (hCardfileInstance,
302:             IDS_SKWGROUP_END_REGARDING, szTime, BUF_LEN);
303:         lpBuf = GlobalLock (lpGroup->hRegarding);
304:         AppendString (lpMessage, szField,
305:             szSpace, lpBuf, szTime,
306:             szReturn, nSize);
307:         GlobalUnlock (lpGroup->hRegarding);
308:     }
309:     // Append Delegate msg
310:     if (lpGroup->wFlags & EVENT_FORWARDED

```

-continued

```

311:     && lpGroup->lpDelegateMsg)
312: {
313:     LoadString    (hCardfileInstance,
314:         IDS_SKWGROUP_START_DELEGATE_MSG,
315:         szField, BUF_LEN);
316:     LoadString    (hCardfileInstance,
317:         IDS_SKWGROUP_END_DELEGATE_MSG,
318:         szTime, BUF_LEN);
319:     if (lstrlen(lpMessage)
320:         + lstrlen(lpGroup->lpDelegateMsg)
321:         + lstrlen(szField) + lstrlen(szTime)
322:         + 10 < nSize)
323:         AppendString (lpMessage, szField,
324:             szSpace, lpGroup->lpDelegateMsg,
325:             szTime, szReturn, nSize);
326: }
327:
328: /*
329:     Append participant's name
330:     and email address to message body.
331: */
332: AppendRecipientInfo (lpMessage, lpGroup,
333:     nSize, nMessageType);
334:
335: /*
336:     Append resource's name
337:     and email address to message body.
338: */
339: if (nMessageType != MSG_REMINDER)
340:     AppendResourceInfo (lpMessage, lpGroup,
341:         nSize, nMessageType);
342:
343: return TRUE;
344: }

```

As its first parameter, the method is passed a pointer to the previously-described GROUP\_APPOINTMENT data structure—the internal buffer storing context information characterizing a group scheduling event (including a message-type). At lines 29–37, local (stack) variables are declared and initialized. At lines 39–40, the method tests whether the message-type involves a resource. For a resource, the method will load a generic header for the message, at line 42. Otherwise (i.e., the “if” statement evaluates to “false”), the method will load into the message buffer, pointed to by lpMessage, a (non-resource) message header, provided that the message-type is Schedule, Reminder, Broadcast, or Reschedule. The comment set forth at lines 54–107 illustrates an exemplary message header. As shown, the message header is formatted in a fashion which permits a user having only basic e-mail support to read the scheduling message and reply (e.g., accept or decline) accordingly.

After the message header has been loaded into the message buffer, the method proceeds to append to the message being constructed the other fields set forth in the previously-described message formats. The method proceeds as follows. At lines 110–114, the method appends the “message-type” delimiter. This is followed by appending the actual message type for the message, at lines 116–160. Specifically, the method first determines the appropriate text string for the message type, at lines 117–158. Then, this message string is appended to the message being constructed, at lines 159–160.

Other fields of the message can be appended to the message being created in a similar manner. At lines 162–176, the method inserts the original date/time in the event that the message is a Reschedule message. At lines 178–192, the method appends the “from” and “to” date and time. At lines 194–199, the method appends the city and country information, and at lines 201–216, the method appends location and time zone information.

At lines 218–225, attribute information is appended. At lines 227–234 and at 236–244, the first and second event IDs

are appended, respectively. In the event that the message indicates that the scheduling event has been forwarded (i.e., delegated) to another user (tested at line 246), the method appends the sender’s name and sender’s e-mail address, at lines 258–265. Additionally, an “append forward” flag is set, at lines 266–271.

At lines 274–291, the method appends the “subject” text. Any “regarding” message is appended at lines 293–308, and any “delegate” message is appended at lines 310–326. To complete the message, the participant’s name and address is appended to the message body, at lines 329–333. Finally, resource information (for any resource) is appended to the message body, at lines 335–341. Now, the message has been successfully completed, and the method may return “true” at line 343. Once the message has been composed, it can be posted to the transport layer, using the previously described Schedule\_Group\_Event Method.

Receipt of the composed scheduling message by a non-SK client without browser support (e.g., WordPerfect Office user) is illustrated in FIGS. 12A–C. FIG. 12A shows an e-mail viewer 1200 displaying the invitation in simple text format 1210. Although the message also includes richer formats (e.g., Attachement(s) 1211), the remote system does not recognize them and, thus, can simply ignore them. As shown in FIG. 12B, the remote user can “reply” to the invitation using his or her own e-mail software, as shown at dialog 1220, and choose to include the message received from the sender, as shown at 1221. Following the instructions in the e-mail invitation, the remote user, as shown in FIG. 12C, edits the reply 1211a by entering an “X” in the [ ] Accept text string at 1213 and entering a brief message between the <BEGIN REPLY MESSAGE> and <END REPLY MESSAGE> delimiters or markers 1217.

The above-illustrated message composition can be extended to automatically generate a Hypertext Markup Language (HTML) form as a scheduling invitation. Here, the HTML form is generated in a conventional manner using HTML code, except that instead of a user hand-crafting the

form, the system automatically maps the scheduling information into appropriate target destinations (fields, buttons, and labels) on the form. For instance, the following data fields are mapped to HTML fields:

HTML field	Mapped information
Message From:	{Initiator}
Date/Time:	{Date/Time}
Duration:	{Duration}
Time Zone:	{Time Zone}
Location:	{Location}
City/Country:	{City}
Subject:	{Subject}
Message:	{Message}

Further, the "Accept" and "Decline" responses are mapped to HTML buttons.

Creation of these control in HTML is straightforward. For instance, the "Accept" and "Decline" buttons can be emitted in HTML format as:

```
<HTML>
...
<BODY>
<FORM>
...
<H1>Accept and Decline Buttons</H1>
<INPUT TYPE="Accept" VALUE="I accept the invitation">
<INPUT TYPE="Decline" VALUE="I decline the invitation">
```

-continued

5

```
</BODY>
</FORM>
...
</HTML>
```

Description of the HTML format is well documented in the trade literature; see e.g., Duncan, R., *An HTML Primer*, PC Magazine, Jun. 13, 1995, pp. 261-270. Further description is provided by Duncan, R., *Publishing HTML Forms on the Web*, PC Magazine, Dec. 5, 1995, pp. 391-403. The disclosures of the foregoing are hereby incorporated by reference.

Receipt of the composed HTML scheduling message by a non-SK client with browser support (e.g., Netscape Navigator user) is illustrated in FIG. 13. The remote user employs the browser 1300 for displaying the invitation in hypertext (HTML) format 1310. The scheduling data fields are automatically placed on the form as HTML form fields 1311. The "Accept" and "Decline" buttons, on the other hand, are placed as buttons 1313. The receiving user can now simply respond to the form, whereupon his or her answer is transmitted back to the sender.

#### d. Methodology for Parsing Messages

Methods of the present invention for identifying and processing incoming group scheduling event messages will now be described. All incoming mail is initially processed by a ReceiveAllIncomingMails method. In an exemplary embodiment, this method may be constructed as follows.

```
1: /*****
2:  This function will receive all incoming group event messages.
3:  *****/
4: int ReceiveAllIncomingMails (void)
5: {
6:     LPMESSAGEINFOSET lpInfoSet;
7:     int i;
8:     nLists = cScheduledLists,
9:     nWantDlg=FALSE,
10:    nResult,
11:    nPaint = FALSE,
12:    nFolderCreated;
13:    char szSender[PATHMAX+1], szTo[PATHMAX+1],
14:          szCC[PATHMAX+1], szBCC[PATHMAX+1],
15:          szSubject[PATHMAX+1], szTime[BUF_LEN+1],
16:          szEmail[65], szEmailType[BUF_LEN+1],
17:          szAttach[PATHMAX+1], szSignature[BUF_LEN+1];
18:    HGLOBAL hMessage;
19:    LPSTR lpMessage;
20:
21:    TurnOnFlashSession ();
22:    lpMessage = AllocLockBuffer (&hMessage, NOTETEXTSIZE * 4);
23:    if (!hMessage)
24:    {
25:        TurnOffFlashSession ();
26:        return FALSE;
27:    }
28:
29:    LoadString (hCardfileInstance,
30:               IDS_SKWGROUP_SIGNATURE, szSignature,
31:               BUF_LEN);
32:
33:    // Get email messages from Exchange
34:    if (!IsDeliverExchange ())
35:    {
36:        char szFolderName[BUF_LEN+1];
37:
38:        // ReceiveAllNewMails (hCardfileWnd);
39:        ReceiveAllMailsNow (hCardfileWnd);
40:
41:        // Create Internet Sidekick Folder.
42:        LoadString (hCardfileInstance,
```

-continued

```

43:         IDS_ISK_FOLDER_NAME, szFolderName, BUF_LEN);
44:     nFolderCreated
45:     = CreateAFirstLevelFolderInDefaultMsgStore
46:     {
47:         hCardfileWnd,
48:         szFolderName
49:     };
50:
51:
52:     // Search for ISK Messages.
53:     lpInfoSet
54:     = SearchSKWScheduleMsgInInboxFolder
55:     {
56:         hCardfileWnd,
57:         szSignature
58:     };
59:
60:
61:     if (!lpInfoSet) // not enough memory
62:     {
63:         UnlockFreeBuffer (hMessage);
64:         TurnOffFlashSession ( );
65:         return FALSE;
66:     }
67:
68:     if (lpInfoSet->ulCount)
69:     {
70:         for (i=0; i<(int)lpInfoSet->ulcount; i++)
71:         {
72:             if (GetMessageContents (hCardfileWnd,
73:                                     lpInfoSet->lpMsgInfo [i].lpEID,
74:                                     szSender, PATHMAX,
75:                                     szEmail, 64,
76:                                     szEmailType, BUF_LEN,
77:                                     szTo, PATHMAX,
78:                                     szCC, PATHMAX,
79:                                     szSubject, PATHMAX,
80:                                     lpMessage, NOTETEXTSIZE * 4,
81:                                     szTime, BUF_LEN,
82:                                     szAttach, PATHMAX))
83:             {
84:
85:                 trim_start_end (szSender, "\", \"");
86:                 trim_start_end (szSender, "\", \"");
87:                 trim_start_end (szEmail, "\", \"");
88:                 trim_start_end (szEmail, "\", \"");
89:                 trim_start_end (szTo, "\", \"");
90:                 trim_start_end (szTo, "\", \"");
91:
92:                 if (!nResult
93:                     = ReceiverMessageHandler (szSender,
94:                                                szEmail, szEmailType,
95:                                                szTo, szCC, szSubject,
96:                                                lpMessage, szAttach,
97:                                                FALSE)))
98:                     continue;
99:
100:                 nPaint = TRUE;
101:                 SetMailRead (hCardfileWnd,
102:                             lpInfoSet->lpMsgInfo[i].lpEID);
103:                 if (!IsLeaveOnServer ())
104:                     DeleteMail (hCardfileWnd,
105:                                 lpInfoSet->lpMsgInfo [i]. lpEID,
106:                                 TRUE);
107:             else
108:             {
109:                 if (nFolderCreated)
110:                     MoveISKMailstoISKMailFolder
111:                     {
112:                         hCardfileWnd,
113:                         lpInfoSet->lpMsgInfo[i].lpEID,
114:                         szFolderName
115:                     };
116:             }
117:         }
118:     }
119: }
120: FreeMessageInfoSet (lpInfoSet);
121: }
122:

```

-continued

```

123: // Get email messages from POP3 account.
124: if (IsDeliverOwnInternet ())
125: {
126:     if (SFMailGetNew (hCardfileWnd, szSignature))
127:     {
128:         if (SFMailBoxOpen ())
129:         {
130:             lpInfoSet = SFMailScanMail (hcardfileWnd);
131:             if (lpInfoSet)
132:             {
133:                 for (i=0; i<(int)lpInfoSet->ulCount; i++)
134:                 {
135:                     if ( SFGetMessageContents
136:                         {
137:                         hCardfileWnd,
138:                         lpInfoSet->lpMsgInfo[i].lpEID,
139:                         szEmail, 64,
140:                         szSender, PATHMAX,
141:                         //szEmailType, BUF_LEN,
142:                         szTo, PATHMAX,
143:                         szCC, PATHMAX,
144:                         szBCC, PATHMAX,
145:                         szSubject, PATHMAX,
146:                         lpMessage, NOTETEXTSIZE * 3,
147:                         szAttach, PATHMAX,
148:                         szTime, BUF_LEN
149:                         }
150:                     }
151:                 }
152:
153:                 trim_start_end (szSender, "\", \"");
154:                 trim_start_end (szSender, "\", \"");
155:                 trim_start_end (szEmail, "\", \"");
156:                 trim_start_end (szEmail, "\", \"");
157:                 trim_start_end (szTo, "\", \"");
158:                 trim_start_end (szTo, "\", \"");
159:
160:                 if (!(nResult
161:                     = ReceiverMessageHandler
162:                     {
163:                     szSender,
164:                     szEmail,
165:                     szEmailType,
166:                     szTo, szCC,
167:                     szSubject, lpMessage,
168:                     szAttach, FALSE
169:                     }))
170:                     continue;
171:                 nPaint = TRUE;
172:                 // delete message from inbox.
173:                 if (nResult)
174:                     SFDeleteMessage
175:                     {
176:                     hCardfileWnd,
177:                     lpInfoSet->lpMsgInfo [i].lpEID
178:                     };
179:                 }
180:             }
181:         }
182:         SFMailBoxClose ();
183:     }
184: }
185: }
186:
187: // unlock and free message block
188: UnlockFreeBuffer (hMessage);
189:
190: TurnOffFlashSession ();
191: if (npaint)
192: {
193:     if (hCoverPageWnd && IsGroupPageFront ())
194:     {
195:         Sort_GroupEventList (hCoverPageWnd, NULL,
196:                               Get_Current_SortFlag ());
197:         if (!nLists)
198:         {
199:             }
200:         }
201:     else
202:         SortGroupEventInfo (hScheduledList, cScheduledLists,

```

```

203:         Get_Current_SortFlag ( );
204:     }
205:     return TRUE;
206: }

```

After initializing local (stack) variables at lines 6–19, the method “turns on” the “flash” session, at line 21. This sets a global flag for preventing re-entry, since this is a timer/interrupt driven process. At lines 22–27, the method allocates a memory block for storing an incoming message. If the allocation fails, the method returns “false” at line 26. At lines 29–31, the method loads from a resource file the signature string (i.e., <ISK>). This string will be used to identify incoming group scheduling events.

Now, the method is ready to retrieve e-mail messages from the user’s in-box. At lines 33–121, the method retrieves messages for Microsoft Exchange and creates a folder for those messages (at lines 41–49). Now, the method searches in that folder for group scheduling event messages—that is, messages containing the identifying signature (at lines 52–58). If at least one group scheduling event message is found (i.e., count is greater than zero at line 68), the method sets up a “for” loop at line 70, for looping through each such message. At lines 72–82, the message contents are extracted by a call to GetMessageContents. At lines 85–90, any trailing spaces are trimmed from this extracted information. As shown, the extracted information includes information about the sender (name), e-mail (address), e-mail-type, “To”, “CC”, subject, message, time, and any attachment.

Further processing is performed by a call to ReceiverMessageHandler, at line 93. The ReceiverMessageHandler method provides appropriate message handling (action), based on the extracted or parsed message information. If the message cannot be handled (i.e., the “if” statement at line 92 evaluates to “false”), then the method loops back for the next message, by executing the “continue” statement at line 98. Otherwise, the message can be handled whereupon the method removes the message from the user’s in-box (lines 104–106), and places it in a group scheduling mail folder (lines 109–115). Thereafter, cleanup is performed at line 120. In the event that the user’s e-mail system is from a POP3 (Internet Post Office Protocol) account, the method executes lines 123–185. Here, the method performs essentially the same task as just described for a Microsoft Exchange mail account.

The method completes its operation by freeing up memory at lines 187–188, turning off the flash session flag at line 190, and repaints the user interface with the new information (if needed) at lines 191–204. Thereafter, all incoming mail has been appropriately processed, and the method may return “true” at line 205.

The message handler itself, ReceiverMessageHandler, may be constructed as follows.

```

1:  /*****
2:   This function will handle incoming group event messages.
3:  *****/
4:  int ReceiverMessageHandler (LPSTR lpFrom,
5:                             LPSTR lpEmail,
6:                             LPSTR lpEmailType,
7:                             LPSTR lpTo,
8:                             LPSTR lpCC,
9:                             LPSTR lpSubject,
10:                            LPSTR lpMessage,
11:                            LPSTR lpAttachFile,
12:                            int nReplyCcode)
13:  {
14:      LPSTR lpShortMsg = NULL, lpNew = NULL;
15:      int nMsgID
16:      = MessageTypeParser (lpSubject, lpMessage, &lpNew),
17:      nReplyType = 0;
18:
19:      if (nMsgID == GROUP_SCHEDULE
20:          || nMsgID == GROUP_BROADCAST
21:          || nMsgID == GROUP_RESCHEDULE)
22:      {
23:          lpShortMsg
24:          = IsThisAutoReplyMessage (lpMessage,
25:                                   &nMsgID,
26:                                   &nReplyType);
27:      }
28:
29:      switch (nMsgID)
30:      {
31:          case GROUP_REMINDER:
32:          case GROUP_SCHEDULE:
33:          case GROUP_BROADCAST:
34:              return Handle_Schedule_Meeting
35:              {
36:                  lpFrom, lpEmail,
37:                  lpEmailType, lpTo,
38:                  lpCC, lpMessage,
39:                  lpAttachFile, nMsgID
40:              };
41:      }

```

-continued

```

42:     case GROUP_REPLY:
43:         return Handle_Reply
44:         {
45:             lpFrom, lpEmail,
46:             lpTo, lpCC,
47:             lpSubject, lpMessage,
48:             nReplyType, lpShortMsg
49:         };
50:
51:     case GROUP_FREETIME:
52:         return CreateFreeTime_Matrix_Send
53:         {
54:             lpFrom, lpEmail,
55:             lpCC, lpMessage
56:         };
57:
58:     case GROUP_RESCHEDULE:
59:         return Handle_Reschedule
60:         {
61:             lpFrom, lpEmail,
62:             lpEmailType, lpTo,
63:             lpCC, lpMessage,
64:             nReplyCode
65:         };
66:
67:     case GROUP_MEETING_NOTE:
68:         return Handle_Meeting_Note
69:         {
70:             lpFrom, lpCC,
71:             lpMessage, lpAttachFile
72:         };
73:
74:     case GROUP_CANCEL_MEETING:
75:         return Handle_Cancel_Meeting (lpFrom, lpCC, lpMessage);
76:
77:     case GROUP_NEW_RESOURCE:
78:         return Handle_AddNewResource (lpFrom,
79:                                     lpEmail, lpMessage);
80:
81:     case GROUP_RESOURCE_SCHEDULE:
82:     case GROUP_RESOURCE_RESCHEDULE:
83:         return Handle_Resource_Schedule
84:         {
85:             lpFrom, lpEmail,
86:             lpEmailType, lpTo,
87:             lpCC, lpMessage
88:         };
89:
90:     case GROUP_RESOURCE_CANCEL:
91:         return Handle_Resource_Cancel (lpFrom,
92:                                     lpEmail, lpMessage);
93:
94:     case GROUP_FREE_RESOURCE:
95:         return Handle_Free_Resource (lpFrom, lpEmail, lpMessage);
96:
97:     case GROUP_TRANSFER_RESOURCE:
98:         return Handle_Transfer_Resource
99:         {
100:             lpFrom, lpEmail,
101:             lpEmailType, lpMessage,
102:             lpAttachFile
103:         };
104:     }
105:     return TRUE;
106: }

```

As shown, the method is invoked with parameters set equal to the just-extracted e-mail information. After declaring local variables at line 14, the method parses out the message type, at lines 15-16. At lines 19-21, the method examines whether the message is one which it can automatically reply to; such a condition holds true when the message is for a Group Schedule, Broadcast, or Reschedule. At lines 20-104, the method establishes a case statement which switches on message type or ID, for invoking more specialized handlers. Group Reminder, Schedule, and Broadcast messages, for instance, invoke a Handle\_Schedule\_Meeting handler at line 34. A group reply

message, on the other hand, invokes a Handle\_Reply handler at line 43. Other handlers are invoked in a corresponding manner. A list of exemplary handlers is appended herewith as Appendix B.

While the invention is described in some detail with specific reference to a single preferred embodiment and certain alternatives, there is no intent to limit the invention to that particular embodiment or those specific alternatives. Thus, the true scope of the present invention is not limited to any one of the foregoing exemplary embodiments but is instead defined by the appended claims.



## APPENDIX A

## Exemplary Message Formats

## Group scheduling format design

## 1. Internet appointment message format

Every type of message will have following two parts at least.

<Product Signature> := <ISK>(or <SIS>)

<Message Type> := <Schedule | Broadcast | Reply | Free Time | Reminder | Reschedule | Meeting Note | Cancel | Recurring | New Resource | Resource | Action | Synchronization>

## Schedule and Broadcast:

<From Date Time> := YY/MM/DD HH:MM  
 <To Date Time> := YY/MM/DD HH:MM  
 <Location> := Place name where the meeting will be held.  
 <City> := City name where the meeting will be held. It is related to Time Zone.  
 <Time Zone> := <Time zone name where appointment time is based on>  
 <Attributes> := <Alarm>  
 <EventID1> := A unique ID for the event.  
 <EventID2> := The second part of the ID.  
 <Subject> := <Message text></Subject>  
 <Message Content> := <Regarding text></Message Content>  
 <Participants> := "Name; Email address; Email Type" "Name; Email Address; Email Type" </Participants>  
 <CC> := "Name; Email address; Email Type" "Name; Email Address; Email Type" </CC>  
 <Room> := "Name; Email address; Email Type" "Name; Email Address; Email Type" </Room>  
 <Equipment> := "Name; Email address; Email Type" "Name; Email Address; Email Type" </Equipment>

If this message is a broadcast message, it requires no ACCEPT response.

## Reply:

<Reply Types> := <Accept | Decline | Reschedule Required | Sleep | Forward | Resource | Free Time>  
 <ForwardTo> := <Name; E-mail address; Email Type></Forward To>  
 << If Reschedule required  
 <From Date Time> := YY/MM/DD HH:MM  
 <To Date Time> := YY/MM/DD HH:MM  
 >>  
 <EventID1> := A unique ID for the event.  
 <EventID2> := The second part of the ID.  
 <Short Message> := <Short text message></Short Message>  
 <<If it is a Reschedule Required, Free Time Request, or Resource Reservation type of Schedule

There is a file attached on this message

The Reschedule Required type of reply is for the purpose of rescheduling right away. For example, a person received a group meeting request. He does not have free time. So he requires to reschedule the meeting with different time.

The purpose of Sleep type is for people on vacation or leaving.

Free Time: (A Free Time request message)

<EventID1> := A unique ID for this Free Time request.  
 <EventID2> := The second part of the ID.  
 <From Date Time> := YY/MM/DD HH:MM  
 <To Date Time> := YY/MM/DD HH:MM  
 <Subject> := <Message text></Subject>

This is an auto reply message request. The reply result of Free Time report is a text matrix with 0 and 1.

## Reschedule:

<Old Date/Time> := YY/MM/DD; HH:MM.  
 <From Date Time> := YY/MM/DD HH:MM  
 <To Date Time> := YY/MM/DD HH:MM  
 <Location> := Place name where the meeting will be held.  
 <City> := City name where the meeting will be held. It is related to Time Zone.  
 <Time Zone> := <Time zone name where appointment time is based on>  
 <Attributes> := <Alarm | Previously Status: Accept; Decline>  
 <EventID1> := A unique ID for the event.  
 <EventID2> := The second part of the ID.  
 <Subject> := <Message text></Subject>  
 <Message Content> := <Regarding text></Subject>  
 <Participants> := "Name; Email address; Email Type" "Name; Email Address; Email Type" </Participants>  
 <CC> := "Name; Email address; Email Type" "Name; Email Address; Email Type" </CC>  
 <Room> := "Name; Email address; Email Type" "Name; Email Address; Email Type" </Room>

## APPENDIX A-continued

## Exemplary Message Formats

<Equipment> := "Name; Email address; Email Type" "Name; Email Address; Email Type" </>  
 </Equipment>

If this meeting is previously accepted, then a forced change appointment list will happen. If this meeting is previously declined or not responded, then it is handled like a normal schedule request.

A Resource request message will be sent again when a Reschedule message is sent.

Reminder:

<From Date Time> := YY/MM/DD HH:MM  
 <To Date Time> := YY/MM/DD HH:MM  
 <Location> := Place name where the meeting will be held.  
 <City> := City name where the meeting will be held. It is related to Time Zone.  
 <Time Zone> := <Time zone name where appointment time is based on>  
 <Attributes> := <Alarm | 1 Previous Status: Accept; Decline>  
 <EventID1> := A unique ID for the event.  
 <EventID2> := The second part of the ID.  
 <Subject> := <Message text></Subject>  
 <Message Content> := <Regarding text></Message Content>

#if not Broadcast

<Participants> := "Name; Email address; Email Type" "Name; Email Address; Email Type" </Participants>  
 <CC>:= "Name; Email address; Email Type" "Name; Email Address; Email Type" </CC>  
 <Room> := "Name; Email address; Email Type" "Name; Email Address; Email Type" </Room>  
 <Equipment> := "Name; Email address; Email Type" "Name; Email Address; Email Type" </Equipment>

If this meeting is previously accepted, the ACCEPT button will be grayed out. If this meeting is not responded, it is handled like a normal schedule.

Meeting Note:

<From Date Time> := YY/MM/DD HH:MM  
 <To Date Time> := YY/MM/DD HH:MM  
 <Location> := Place name where the meeting was held.  
 <City> := City name where the meeting will be held. It is related to Time Zone.  
 <Time Zone> := <Time zone name where appointment time is based on>  
 <EventID1> := A unique ID for the event.  
 <EventID2> := The second part of the ID.  
 <Subject> := <Message text></Subject>  
 <Meeting Note> := The meeting note text</Meeting Note>

Cancel a group meeting:

<EventID1> := A unique ID for the event.  
 <EventID2> := The second part of the ID.  
 <Subject> := <Message text></Subject>  
 <Short Message> := <Message text></Short Message>

Resource: (Resource Reservation Message)

<From Date Time> := YY/MM/DD HH:MM  
 <To Date Time> := YY/MM/DD HH:MM  
 <EventID1> := A unique ID for the event  
 <EventID2> := The second part of the ID  
 <Subject> := <Message text></Subject>  
 <Message Content> := <Regarding text></Message Content>

Schedule a recurring Appointment:

<Start Date> := YY/MM/DD  
 <End Date> := YY/MM/DD  
 <From Date Time> := YY/MM/DD HH:MM  
 <To Date Time> := YY/MM/DD HH:MM  
 <Recur Pattern 1> := Month | Week | Day  
 <Recur Pattern 2> := Frequency  
 <Recur Pattern 3> := Include and Exclude  
 <Location> := Place name where the meeting will be held.  
 <City> := City name where the meeting will be held. It is related to Time Zone.  
 <Time Zone> := <Time zone name where appointment time is based on>  
 <Attributes> := <Alarm>  
 <Subject> := <Message text></Subject>  
 <Message Content> := <Regarding text></Message Content>

Executing an Action cross the serverless network:

<From Date Time> := YY/MM/DD HH:MM  
 <To Date Time> := YY/MM/DD HH:MM  
 <Location> := Place name where the meeting will be held.

## APPENDIX A-continued

## Exemplary Message Formats

---

<City> := City name where the meeting will be held. It is related to Time Zone.  
 <Time Zone> := <Time zone name where appointment time is based on>  
 <EventID1> := A unique ID for the event.  
 <EventID2> := The second part of the ID.  
 <Subject> := <Message text></Subject>  
 <Message Content> := <Regarding text></Message Content>  
 <Action Item> := <Java applet | An executable program | operations which can be  
 recognized in Sidekick>  
 Synchronizing files cross the network:

---

<From Date Time> := YY/MM/DD HH:MM  
 <To Date Time> := YY/MM/DD HH:MM  
 <Location> := Place name where the meeting will be held.  
 <City> := City name where the meeting will be held. It is related to Time Zone.  
 <Time Zone> := <Time zone name where appointment time is based on>  
 <EventID1> := A unique ID for the event.  
 <EventID2> := The second part of the ID.  
 <Subject> := <Message text></Subject>  
 <Message Content> := <Regarding text></Message Content>  
 <File Name>:= <File name which needs to be synchronized>  
 Resource scheduling requires non-overlapped time and a subject. The scheduling is a  
 non-negotiable process. The reply only has two options: Accept and Decline.

---

## APPENDIX B

## Exemplary Message Handlers for Actions

---

```

/*****
*****
int Handle_Schedule_Meeting (LPSTR lpFrom,
                             LPSTR lpEmail,
                             LPSTR lpEmailType,
                             LPSTR lpTo,
                             LPSTR lpCC,
                             LPSTR lpMessage,
                             LPSTR lpAttachFile,
                             int nMsgID)
{
    // This function will scan the incoming message and get all the
    // information related to this event.
    // Information includes following parts:
    // - start date/time and end date/time
    // - time zone
    // - location
    // - City and country
    // - Event properties, such as alarm, tentative, RSVP, etc..
    // - Participants info
    // - Resource allocated for this event
    // - Event ID
    // - Subject
    // - Message body
    // After getting the information, it will add one item into Event List
    // database
    // This function will also handle the Reminder and Confirm message type
    //
    return TRUE;
}
/*****
*****
int Handle_Reschedule (LPSTR lpFrom, LPSTR lpEmail, LPSTR lpEmailType, LPSTR
lpTo, LPSTR lpCC, LPSTR lpMessage, int nReplyCode)
{
    // This function will scan incoming message and get all the information
    // It will set new value to the same item in the Event database
}
/*****
*****
int Handle_Reply (LPSTR lpFrom, LPSTR lpEmail, LPSTR lpTo, LPSTR lpCC, LPSTR
lpSubject, LPSTR lpMessage, int nInReplyType, LPSTR lpShortMsg)
{
    1. Get group schedule information included in lpMessage;
    2. Get Reply type information included in lpMessage;
    3. Handle Group Meeting update, including change recipient status,
    change recipient if it is a forward to type of reply.
  
```

---

## APPENDIX B-continued

## Exemplary Message Handlers for Actions

```

}
/*.....*/
int CreateFreeTime_Matrix_Send (LPSTR lpFrom, LPSTR lpEmail, LPSTR lpCC, LPSTR
lpMessage)
{
    // Create a free time calendar and send to initiator
}
/*.....*/
int Handle_Meeting_Note (LPSTR lpFrom, LPSTR lpCC, LPSTR lpMessage, LPSTR
lpAttachFile)
{
    // get minutes of meeting and attached file and
    // save to the Event
}
/*.....*/
int Handle_Cancel_Meeting (LPSTR lpFrom, LPSTR lpCC, LPSTR lpMessage)
{
    // get information from message
    // delete event from calendar
    // set current event to canceled status
}
/*.....*/
int Handle_Resource_Schedule (LPSTR lpFrom, LPSTR lpEmail, LPSTR lpEmailType,
LPSTR lpTo, LPSTR lpCC, LPSTR lpMessage)
{
    // get information from message
    // try to reserve the resource
    // send reply back to initiator (accept or decline)
}
/*.....*/
int Handle_Resource_Cancel (LPSTR lpFrom, LPSTR lpEmail, LPSTR lpMessage)
{
    // cancel resource reservation
}
/*.....*/
int Handle_Free_Resource (LPSTR lpFrom, LPSTR lpEmail, LPSTR lpMessage)
{
    // create resource free time calendar and send to initiator
}
/*.....*/
int Handle_Transfer_Resource (LPSTR lpFrom, LPSTR lpEmail, LPSTR lpEmailType,
LPSTR lpMessage, LPSTR lpAttachFile)
{
    // get transfer resource and add an item to event database
}

```

## What is claimed is:

1. In a computerized scheduling system, a method for assisting a user with scheduling calendar events in an electronic calendar, the method comprising:

- (a) receiving input from the user specifying an event to schedule together with a lists of participants desired to participate in the event, at least some of said participants employing remote computer systems which are unable to interpret proprietary scheduling formats of the computerized scheduling system of the user;
- (b) in response to said input, generating an electronic scheduling invitation which invites the participants to the event, said scheduling invitation being encoded in a plurality of different message formats, each message format supporting a different level of information content, said plurality of different message formats selected from a group comprising at least a proprietary scheduling format, a Hypertext Markup Language

50

(HTML) format, and a simple electronic-mail format so that said scheduling invitation may be encoded for appropriate processing by disparate remote computer systems including those which are unable to interpret proprietary scheduling formats of the computerized scheduling system of the user;

- (c) sending said scheduling invitation to each participant;
- (d) upon receiving said scheduling invitation, generating an electronic scheduling reply by:
  - (i) decoding the message format having the highest level of information content suitable for the computer system employed by said each participant,
  - (ii) creating an electronic scheduling reply suitable for automatic processing by said computerized scheduling system of the user, said reply including a response indicating whether said each participant can participate in the event, and
  - (iii) sending said scheduling reply to said user; and

63

(e) upon receiving each participant's scheduling reply, automatically updating the calendar based on the response contained within the scheduling reply.

2. In a computerized scheduling system, a method for assisting a user with scheduling calendar events in an electronic calendar, the method comprising:

(a) receiving input from the user specifying an event to schedule together with a lists of participants desired to participate in the event, at least some of said participants employing remote computer systems which are unable to interpret proprietary scheduling formats of the computerized scheduling system of the user;

(b) in response to said input, generating an electronic scheduling invitation which invites the participants to the event, said scheduling invitation being encoded in a plurality of different message formats, each message format supporting a different level of information content;

(c) sending said scheduling invitation to each participant;

(d) upon receiving said scheduling invitation, generating an electronic scheduling reply by:

(i) decoding the message format having the highest level of information content suitable for the computer system employed by said each participant.

(ii) creating an electronic scheduling reply suitable for automatic processing by said computerized scheduling system of the user, said reply including a response indicating whether said each participant can participate in the event, and

(iii) sending said scheduling reply to said user; and

(e) upon receiving each participant's scheduling reply, automatically updating the calendar based on the response contained within the scheduling reply, wherein at least one message format comprises a proprietary scheduling format of the computerized scheduling system of the user.

3. The method of claim 2, wherein said message format which comprises a proprietary scheduling format of the computerized scheduling system of the user is transmitted in the electronic scheduling message as a binary attachment.

4. The method of claim 3, wherein said binary attachment comprises a Multipurpose-Internet-Mail-Extensions binary attachment.

5. The method of claim 1, wherein at least one message format comprises a simple text message ensuring that a participant employing a computer system providing only simple electronic-mail support can reply to said electronic scheduling message.

6. The method of claim 1, wherein said input received from the user comprises an event time and location.

7. The method of claim 6, wherein at least one of the scheduling replies comprises a reply indicating that a participant cannot participate in the event and further includes information indicating an alternate time for scheduling the event.

8. In a computerized scheduling system, a method for assisting a user with scheduling calendar events in an electronic calendar, the method comprising:

(a) receiving input from the user specifying an event to schedule together with a lists of participants desired to participate in the event, at least some of said participants employing remote computer systems which are unable to interpret proprietary scheduling formats of the computerized scheduling system of the user;

(b) in response to said input, generating an electronic scheduling invitation which invites the participants to

64

the event, said scheduling invitation being encoded in a plurality of different message formats, each message format supporting a different level of information content;

(c) sending said scheduling invitation to each participant;

(d) upon receiving said scheduling invitation, generating an electronic scheduling reply by:

(i) decoding the message format having the highest level of information content suitable for the computer system employed by said each participant,

(ii) creating an electronic scheduling reply suitable for automatic processing by said computerized scheduling system of the user, said reply including a response indicating whether said each participant can participate in the event, and

(iii) sending said scheduling reply to said user; and

(e) upon receiving each participant's scheduling reply, automatically updating the calendar based on the response contained within the scheduling reply, wherein said message format which comprises a Hypertext Markup Language (HTML) form automatically generated by the system based on said input from the user, said HTML form capable of being viewed by any participant having an HTML browser.

9. The method of claim 8, wherein said HTML form comprises form fields displaying said input from the user, and further comprises screen buttons which a participant can select for generating a reply.

10. In a computerized scheduling system, a method for assisting a user with scheduling calendar events in an electronic calendar, the method comprising:

(a) receiving input from the user specifying an event to schedule together with a lists of participants desired to participate in the event, at least some of said participants employing remote computer systems which are unable to interpret proprietary scheduling formats of the computerized scheduling system of the user;

(b) in response to said input, generating an electronic scheduling invitation which invites the participants to the event, said scheduling invitation being encoded in a plurality of different message formats, each message format supporting a different level of information content;

(c) sending said scheduling invitation to each participant;

(d) upon receiving said scheduling invitation, generating an electronic scheduling reply by:

(i) decoding the message format having the highest level of information content suitable for the computer system employed by said each participant,

(ii) creating an electronic scheduling reply suitable for automatic processing by said computerized scheduling system of the user, said reply including a response indicating whether said each participant can participate in the event, and

(iii) sending said scheduling reply to said user; and

(e) upon receiving each participant's scheduling reply, automatically updating the calendar based on the response contained within the scheduling reply, wherein said a different level of information content comprises selected ones of a proprietary scheduling format of the computerized scheduling system, a Hypertext Markup Language (HTML) format, and a simple text format.

11. The method of claim 1, wherein said electronic scheduling invitation includes an embedded identifier which is incorporated into each reply so that said computerized

scheduling system can automatically differentiate the replies from other electronic mail which the user receives.

12. The method of claim 11, wherein said embedded identifier includes information allowing said computerized scheduling system to automatically identify the replies as being associated with a particular electronic scheduling invitation.

13. The method of claim 1, wherein each reply includes a participant identifier allowing said computerized scheduling system to automatically associate each reply with a particular participant.

14. The method of claim 13, wherein step (c) includes indicating in the calendar which participants can attend.

15. The method of claim 1, wherein step (c) includes:

transmitting said scheduling invitation via a particular electronic-mail format of America On-line, CompuServe, Microsoft Exchange, and Internet POP3, wherein the particular electronic-mail format employed is selected based on an electronic-mail address for each participant.

16. An automated electronic scheduling system comprising:

a computer having a processor and a memory;  
a user interface for inputting a particular event for scheduling;

a composer, responsive to said user interface, for automatically generating an electronic mail (e-mail) invitation which encodes scheduling information in formats of differing levels of information content for supporting e-mail systems other than said automated electronic scheduling system, said plurality formats selected from a group comprising at least a proprietary scheduling format, a Hypertext Markup Language (HTML) format, and a simple electronic-mail format so that said e-mail invitation may be encoded for appropriate processing by disparate e-mail systems including those which are unable to interpret proprietary scheduling formats of said automated electronic scheduling system;

an e-mail transport mechanism for sending the e-mail invitation to each desired participants, at least some of who reply to the e-mail invitation; and  
means for identifying each reply which responds to the e-mail invitation; and

a parser, responsive to identification means, for extracting from each reply information indicating whether a desired participant can attend the particular event.

17. The system of claim 16, further comprising:

a calendar update module for automatically indicating which desired participants can attend the particular event.

18. The system of claim 16, wherein said composer inserts an identifier into the e-mail invitation so that any reply which incorporates contents of the e-mail invitation can be associated with the particular event.

19. The system of claim 18, wherein said identifier is inserted into a subject line of the e-mail invitation.

20. The system of claim 16, wherein said composer inserts into the e-mail invitation for each particular desired participant an identifier for the participant so that any reply which incorporates contents of the e-mail invitation can be associated with the particular desired participant.

21. An automated electronic scheduling system comprising:

a computer having a processor and a memory;  
a user interface for inputting a particular event for scheduling;

a composer, responsive to said user interface, for automatically generating an electronic mail (e-mail) invitation which encodes scheduling information in formats of differing levels of information content for supporting e-mail systems other than said automated electronic scheduling system;

an e-mail transport mechanism for sending the e-mail invitation to each desired participants, at least some of who reply to the e-mail invitation; and

means for identifying each reply which responds to the e-mail invitation; and

a parser, responsive to identification means, for extracting from each reply information indicating whether a desired participant can attend the particular event, wherein said formats comprise at least a simple text message understood by any participant having a system with simple e-mail support, a Hypertext Markup Language (HTML) form understood by any participant having a system with an Internet browser, and a binary attachment understood by any participant having the automated electronic scheduling system.

22. The system of claim 21, wherein said simple text message includes delimiters for recording a desired participant's response in a manner which can be identified by the parser.

23. The system of claim 22, wherein said delimiters comprise text strings having a form substantially like [ ] Accept and [ ] Decline, for allowing a desired participant to indicate whether the participant can attend the particular event.

24. The system of claim 22, wherein said delimiters further comprise text strings having a form substantially like <BEGIN REPLY MESSAGE> and <END REPLY MESSAGE>, for allowing a desired participant to enter a reply which is automatically recognized and processed by the system.

25. An automated electronic scheduling system comprising:

a computer having a processor and a memory;  
a user interface for inputting a particular event for scheduling;

a composer, responsive to said user interface, for automatically generating an electronic mail (e-mail) invitation which encodes scheduling information in formats of differing levels of information content for supporting e-mail systems other than said automated electronic scheduling system;

an e-mail transport mechanism for sending the e-mail invitation to each desired participants, at least some of who reply to the e-mail invitation; and

means for identifying each reply which responds to the e-mail invitation; and

a parser, responsive to identification means, for extracting from each reply information indicating whether a desired participant can attend the particular event, wherein said e-mail transport mechanism is a selected one of a Microsoft MAPI (Messaging Application Programming Interface) compliant e-mail transport mechanism and a POP3 (Internet Post Office Protocol) compliant e-mail transport mechanism.

26. A method for unattended scheduling of resources, the method comprising:

storing in a computer information describing a set of resources which are available for use by individuals;  
providing at least one electronic mail (e-mail) account at the computer for receiving e-mail scheduling requests

67

from the individuals for scheduling use of the resources at particular times;  
 for such an e-mail request received at the computer, processing the e-mail request for identifying a particular individual who is requesting a particular resource at a requested time;  
 comparing the received request against a scheduling calendar listing availability of the particular resource at the requested time; and  
 if the particular resource is available at the requested time, automatically sending a reply e-mail message to the particular individual confirming acceptance of the scheduling request and updating the scheduling calendar for indicating that the particular resource is now scheduled at the requested time for use by the particular individual.  
 27. The method of claim 26, further comprising:  
 if the particular resource is unavailable at the requested time, automatically sending a reply e-mail message to the particular individual denying acceptance of the scheduling request.  
 28. The method of claim 27, wherein said reply e-mail message further comprises a report indicating availability of the particular resource for a given time period.  
 29. The method of claim 26, wherein said set of resources include at least two types of resources selected from a movable type of resource and an immovable type of resource.

68

30. The method of claim 29, wherein said movable type of resource includes an "equipment" resource type.  
 31. The method of claim 29, wherein said immovable type of resource includes a "room" resource type.  
 32. The method of claim 31, wherein said scheduling calendar also maintains schedules for meetings of the individuals and wherein individuals are not allowed to schedule two meetings which require an identical resource which is of type "room."  
 33. The method of claim 29, wherein a resource which is immovable is not allowed to accept scheduling requests specifying times which overlap.  
 34. The method of claim 29, wherein a resource which is movable is allowed to accept scheduling requests specifying times which overlap.  
 35. The method of claim 26, wherein said processing the e-mail request includes parsing the e-mail request to extract an identifier indicating that the e-mail is a scheduling request for a resource.  
 36. The method of claim 26, wherein said requested time is automatically converted by the system to a time zone relative to where the particular individual resides.

\* \* \* \* \*



US006466236B1

(12) **United States Patent**  
**Pivowar et al.**

(10) Patent No.: **US 6,466,236 B1**  
 (45) Date of Patent: **Oct. 15, 2002**

(54) **SYSTEM AND METHOD FOR DISPLAYING  
 AND MANIPULATING MULTIPLE  
 CALENDARS ON A PERSONAL DIGITAL  
 ASSISTANT**

5,809,242 A 9/1998 Shaw et al. .... 709/217

(List continued on next page.)

#### FOREIGN PATENT DOCUMENTS

WO WO99/06900 11/1999

#### OTHER PUBLICATIONS

Puma Technology, Intellisync, <http://www.pumatech.com/intellisync.html>, Feb. 22, 1999.

TrueSync Technology, TrueSync Tehcnology Platform, <http://www.starfish.com/products/truetech/truetech.html>, Feb. 22, 1999.

When.com, What is When.com?, <http://www.when.com>, Apr. 7, 1999.

PointCast, PointCast Network, <http://www.pointcast.com/products/pcn/index.html?homepb>, Apr. 7, 1999.

PointCast, PointCast Network, <http://www.pointcast.com/products/pcn/hwork.html?pcnidxbdy> Apr. 7, 1999.

Primary Examiner—Crescelle N. dela Torre

#### (57) ABSTRACT

A portable, hand-held personal digital assistant is provided for simultaneously depicting multiple calendars on a single display. The personal digital assistant includes a portable, hand-held housing including a top face, a bottom face, and a side wall therebetween for defining an interior space. An input device is situated on the top face of the housing for allowing input of data. Associated therewith is a display situated on the top face of the housing for depicting data. Situated in the interior space of the housing is memory for storing a plurality of calendars each including a plurality of scheduled matters. Finally, controller, is situated in the interior space of the housing and connected between the input device, the display, and the memory. The controller serves for simultaneously depicting a plurality of the calendars on the display. By conveniently displaying the multiple calendars, the present invention allows a user to more effectively manipulate the same.

27 Claims, 17 Drawing Sheets

(75) Inventors: **Alvin Pivowar; Steve Hanrahan; Pete Grillo**, all of Portland, OR (US)

(73) Assignee: **Palm, Inc.**, Santa Clara, CA (US)

(\*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/288,774**

(22) Filed: **Apr. 8, 1999**

(51) Int. Cl.<sup>7</sup> ..... **G06F 3/00**

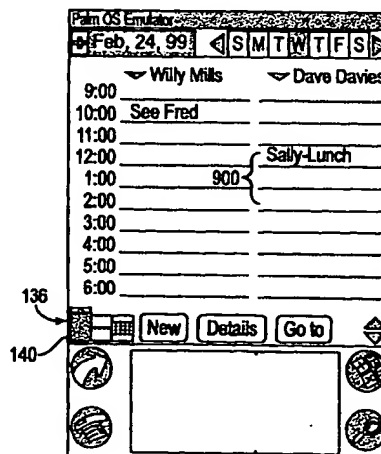
(52) U.S. Cl. .... **345/835; 345/864; 345/963**

(58) Field of Search ..... **345/326, 329, 345/333, 334, 339, 340, 348, 350, 351, 352, 354, 357, 963, 169, 173, 700, 703, 733, 744, 764, 765, 775, 776, 781, 810, 835, 840, 843, 864, 854, 866; 705/8, 9**

#### (56) References Cited

##### U.S. PATENT DOCUMENTS

4,831,552 A	• 5/1989	Scully et al. ....	345/329
5,129,057 A	• 7/1992	Strope et al. ....	345/348
5,214,768 A	5/1993	Martin et al. ....	711/114
5,261,045 A	11/1993	Scully ....	345/751
5,412,791 A	5/1995	Martin et al. ....	711/114
5,457,476 A	• 10/1995	Jenson ....	345/146
5,479,411 A	12/1995	Klein ....	379/88.13
5,528,745 A	• 6/1996	King et al. ....	345/326
5,557,659 A	9/1996	Hyde-Thompson ....	379/88.13
5,572,643 A	11/1996	Judson ....	709/218
5,621,458 A	• 4/1997	Mann et al. ....	348/232
5,647,002 A	7/1997	Brunson ....	709/206
5,684,990 A	11/1997	Boothby ....	707/203
5,740,549 A	4/1998	Reilly et al. ....	705/14
5,745,884 A	4/1998	Carnegie et al. ....	705/34
5,790,974 A	8/1998	Tognazzini ....	701/204





## U.S. PATENT DOCUMENTS

5,862,346 A	1/1999	Kley .....	709/245	6,023,708 A	2/2000	Mendez .....	707/203
5,870,759 A	2/1999	Bauer et al. ....	707/201	6,026,369 A	2/2000	Capek .....	705/14
5,877,759 A *	3/1999	Bauer .....	345/339	6,026,371 A	2/2000	Beck et al. ....	705/14
5,907,678 A	5/1999	Housel, III et al. ....	709/213	6,034,621 A	3/2000	Kaufman .....	340/7.21
5,933,811 A	8/1999	Angles et al. ....	705/14	6,034,661 A	3/2000	Servan-Scheiber et al. .	345/668
5,949,975 A	9/1999	Batty et al. ....	709/213	6,034,683 A *	3/2000	Mansour et al. ....	345/339
5,966,714 A	10/1999	Huang .....	707/201	6,058,415 A	5/2000	Polcyn .....	709/200
5,982,891 A	11/1999	Ginter et al. ....	705/54	6,101,480 A *	8/2000	Conmy et al. ....	705/9
5,999,912 A	12/1999	Wodarz et al. ....	705/14	6,131,096 A	10/2000	Ng .....	707/10
6,000,000 A	12/1999	Hawkins et al. ....	707/201	6,131,116 A	10/2000	Riggins .....	709/219
6,009,410 A	12/1999	LeMole et al. ....	705/14	6,138,245 A	10/2000	Son et al. ....	713/400
6,011,537 A	1/2000	Slotznick .....	345/733	6,151,606 A	11/2000	Mendez .....	707/201
6,014,502 A	1/2000	Moraes .....	709/219	6,161,146 A	12/2000	Kley .....	709/248

\* cited by examiner

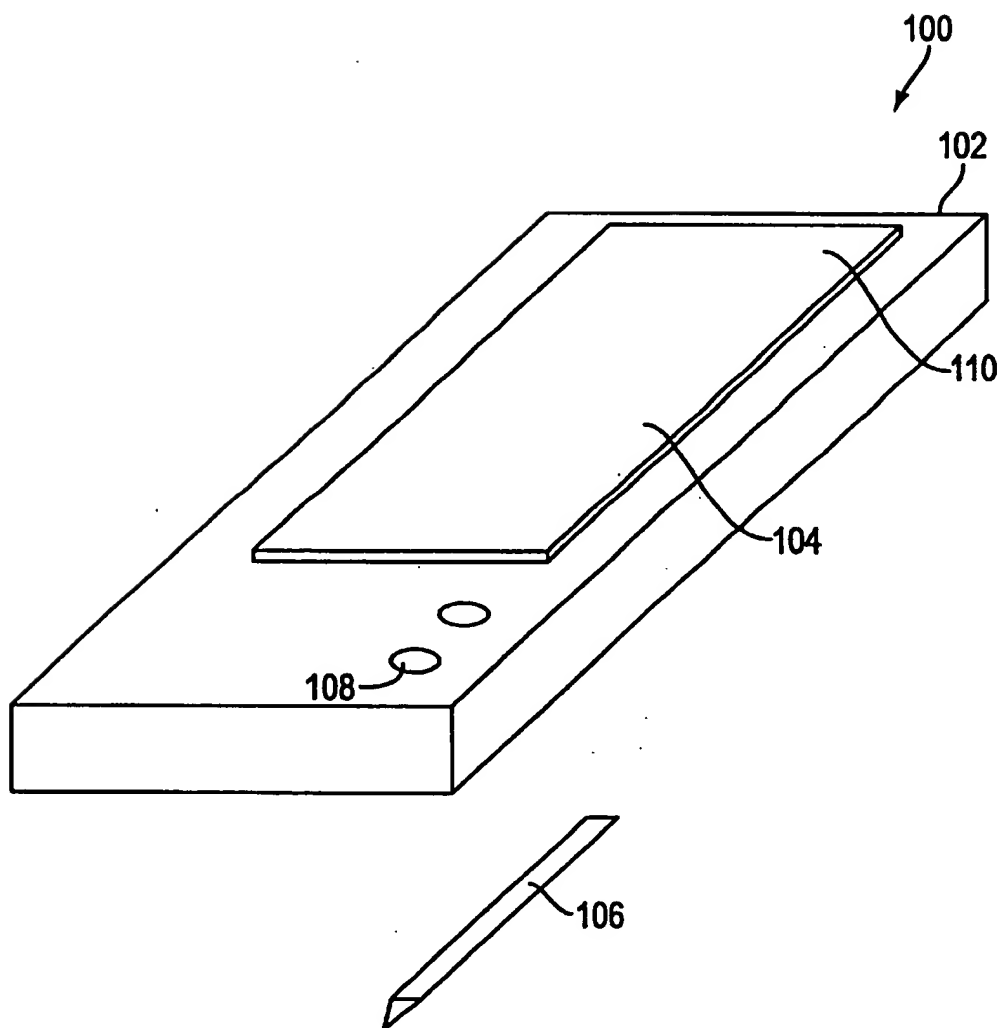


FIG. 1

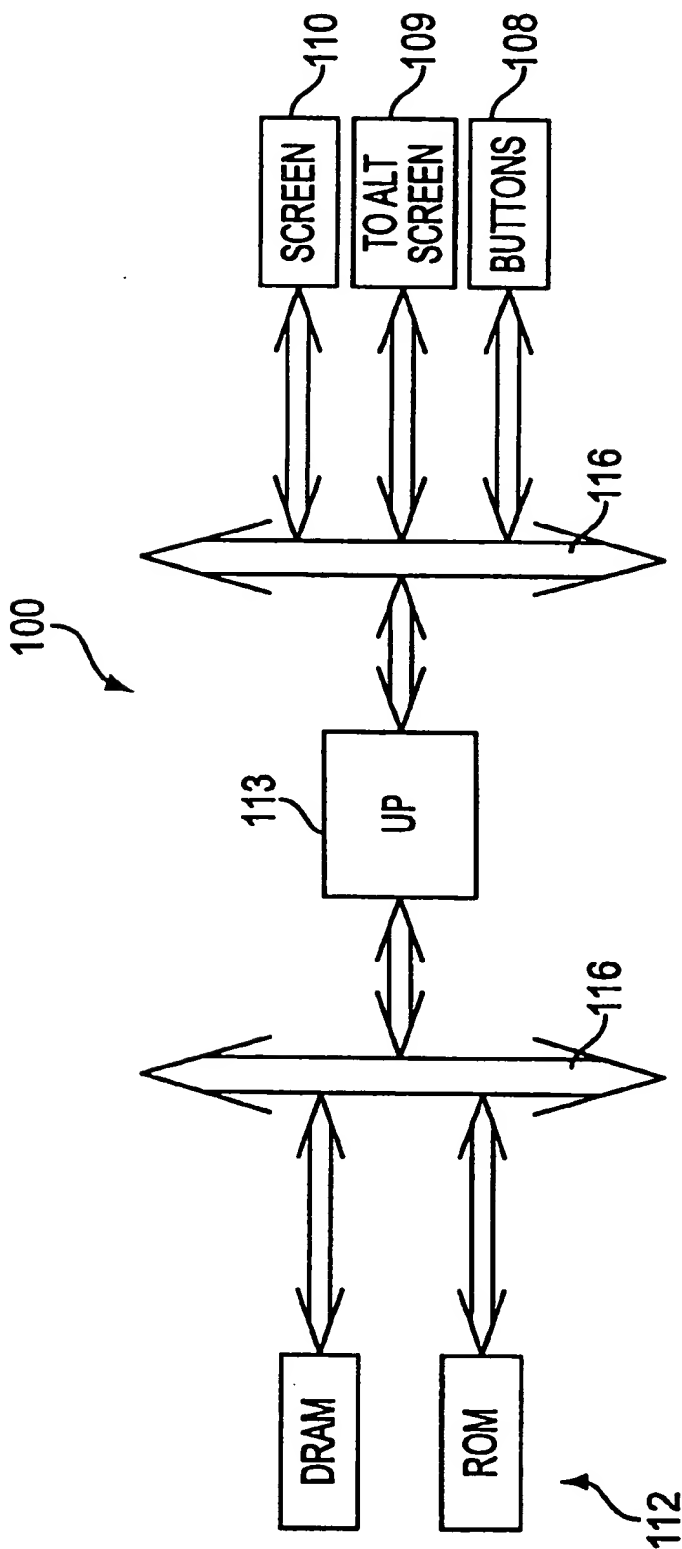


FIG. 2

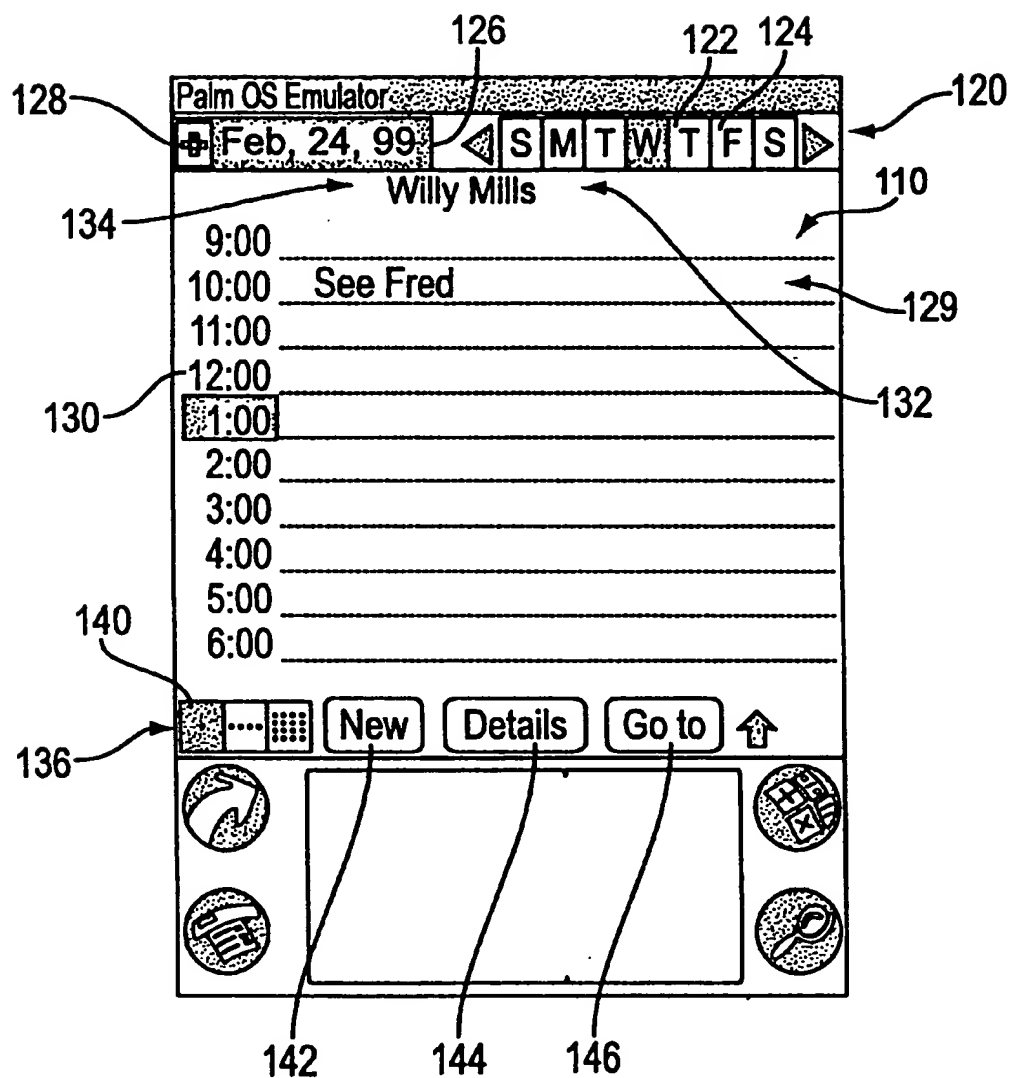
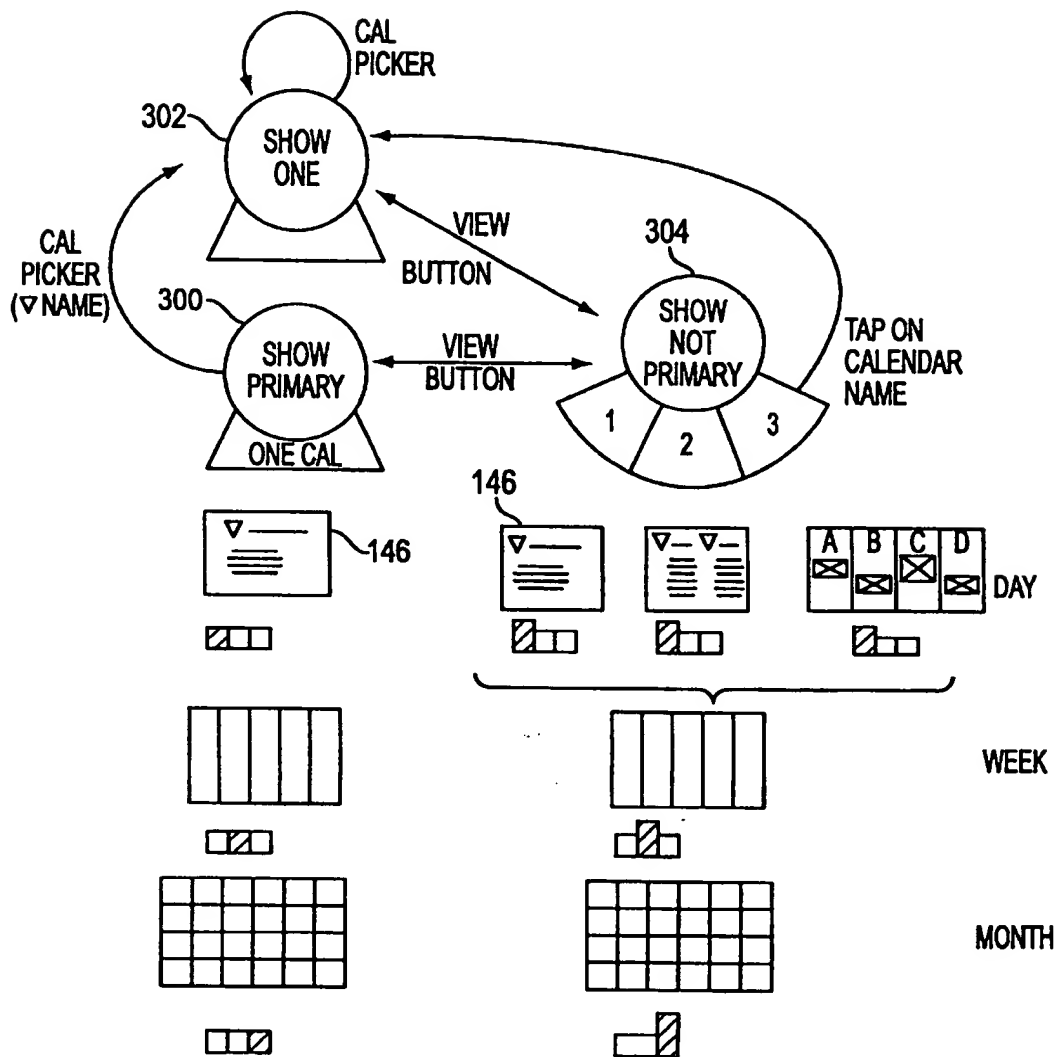


FIG. 3A



**FIG. 3B**

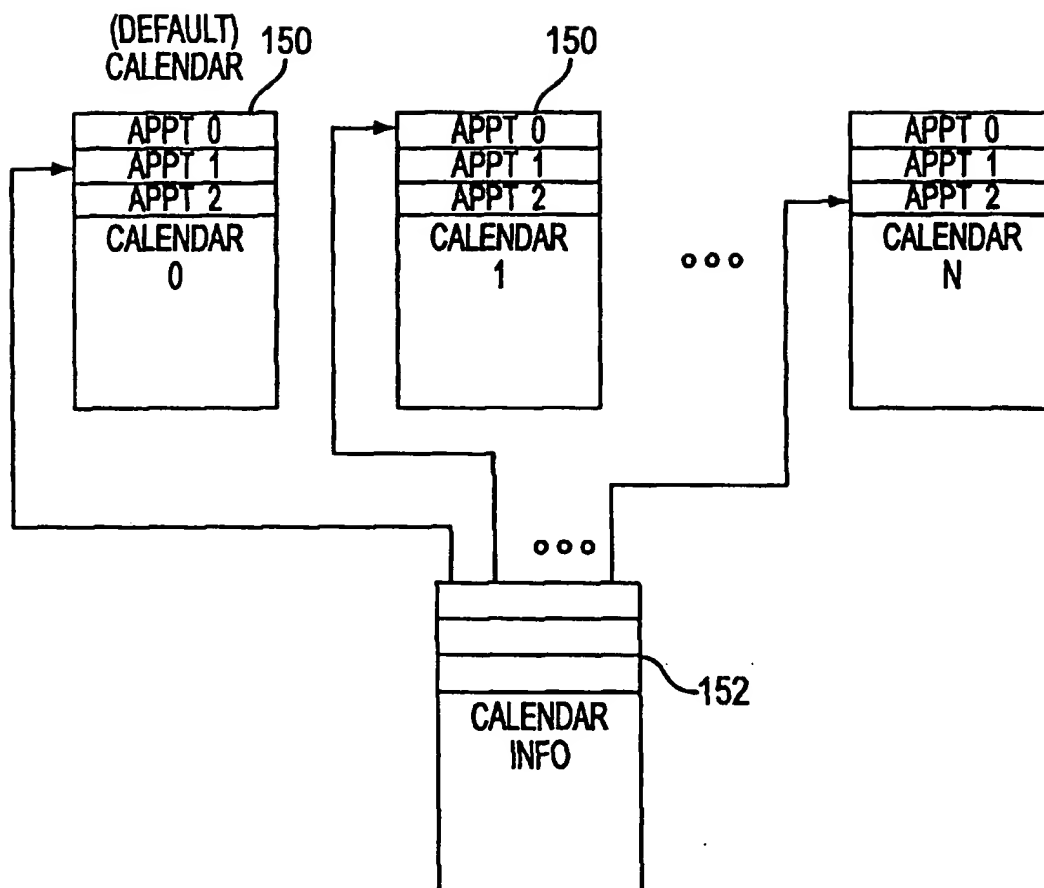


FIG. 3C

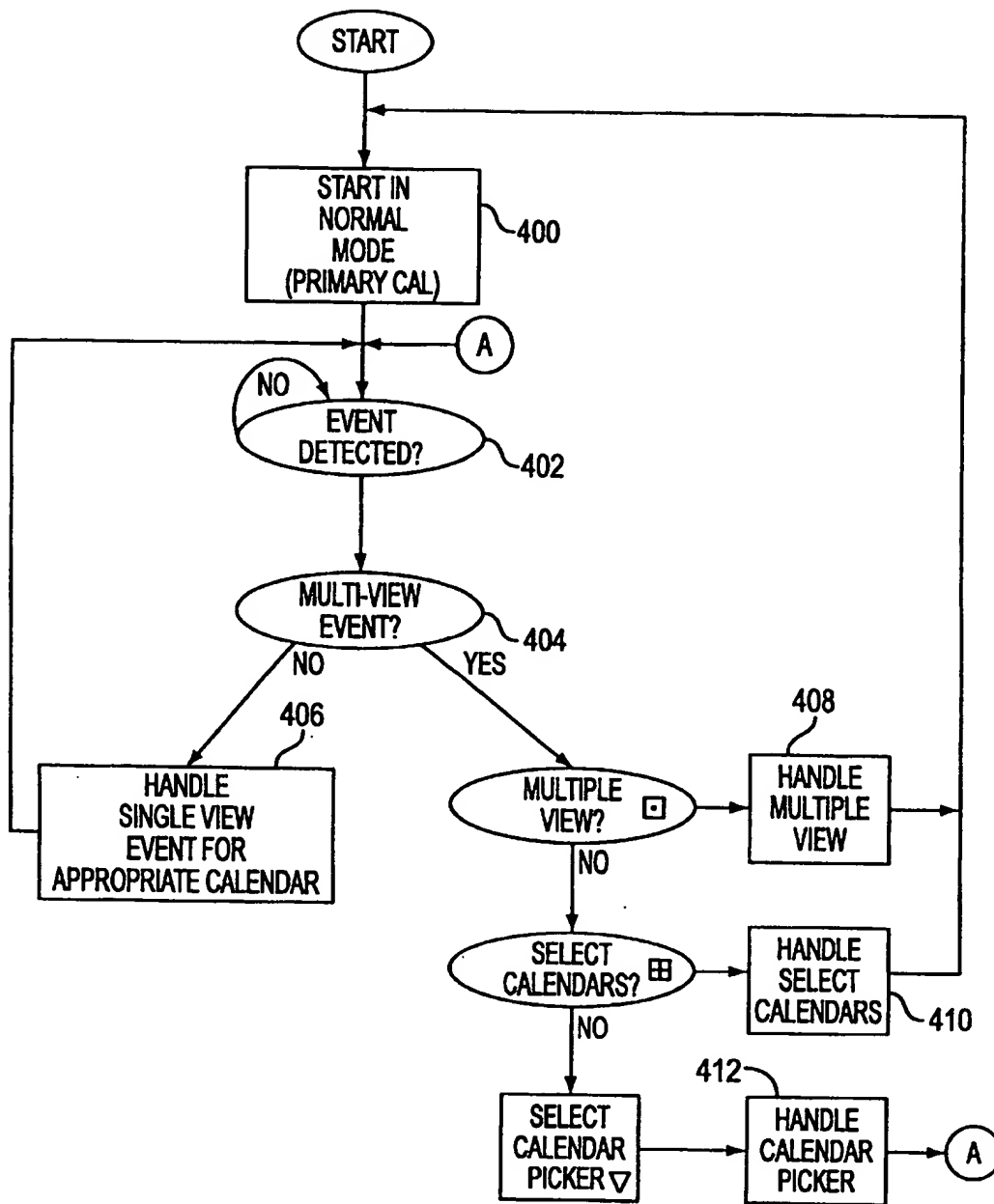


FIG. 4

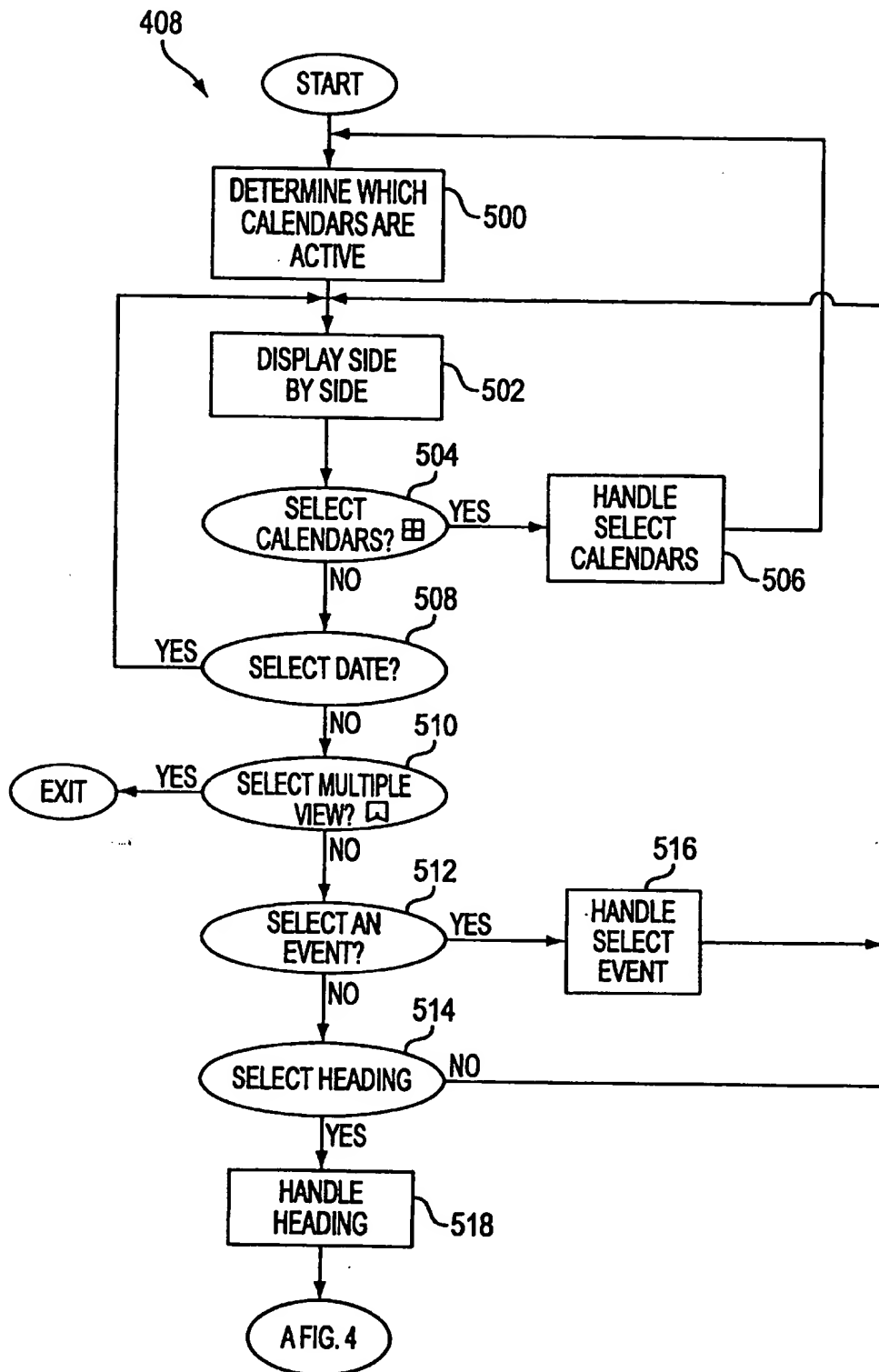


FIG. 5



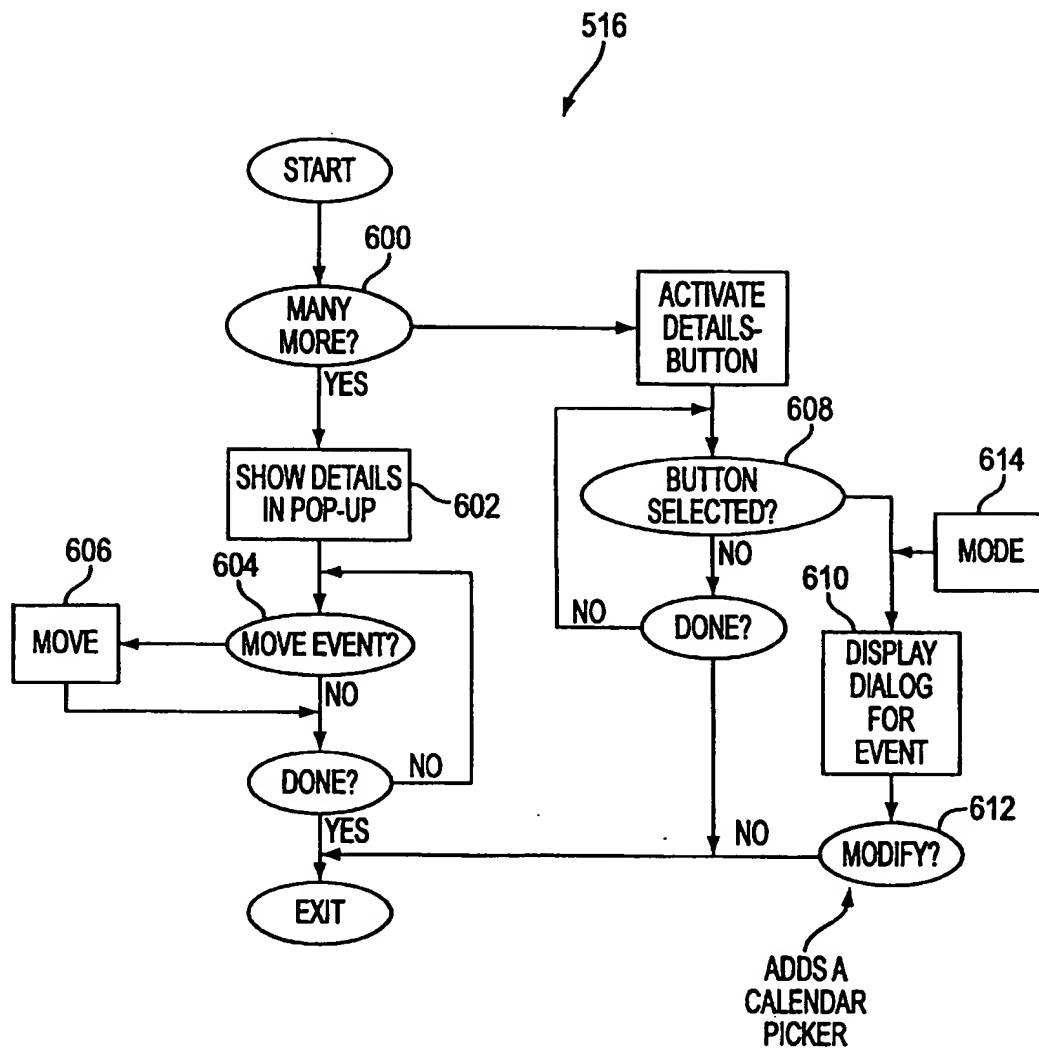


FIG. 6

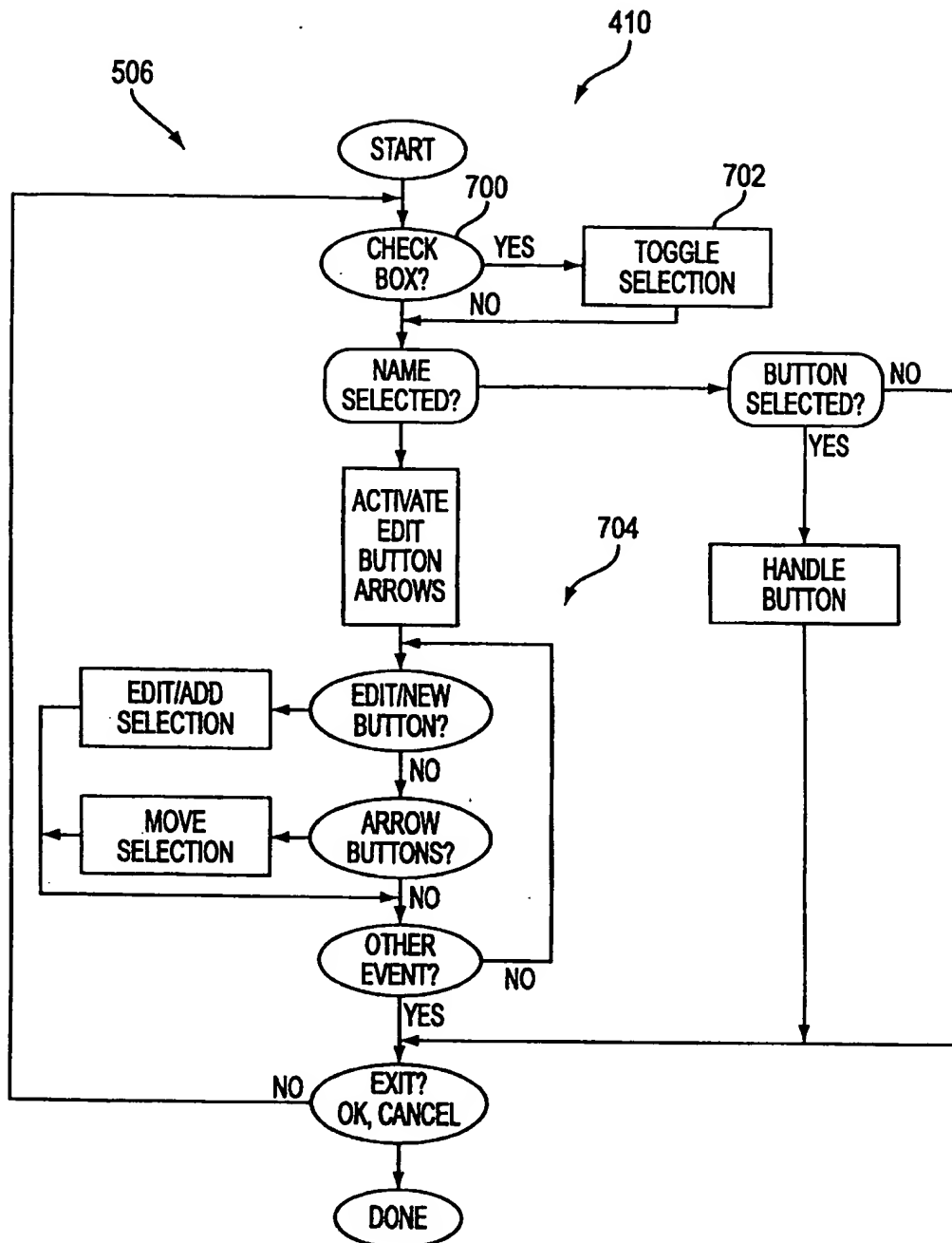


FIG. 7

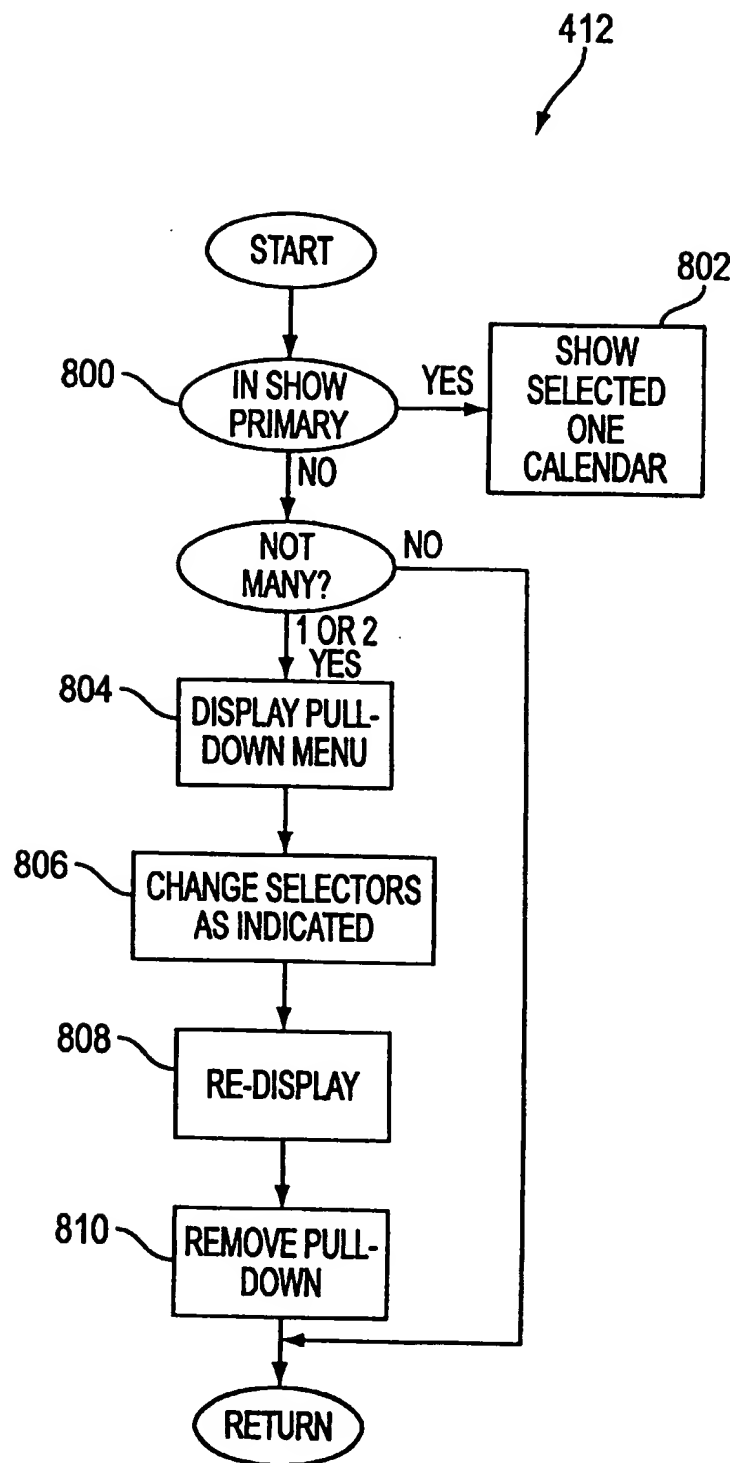


FIG. 8

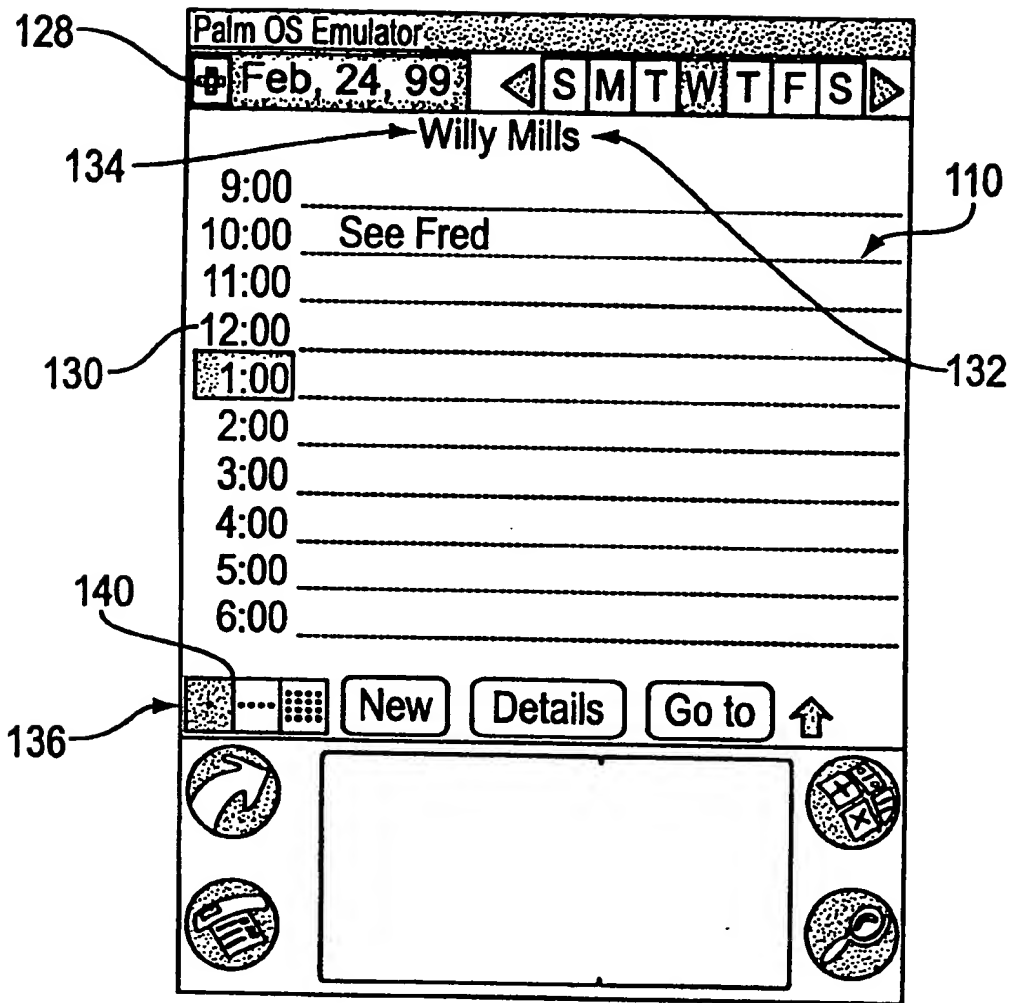


FIG. 9A

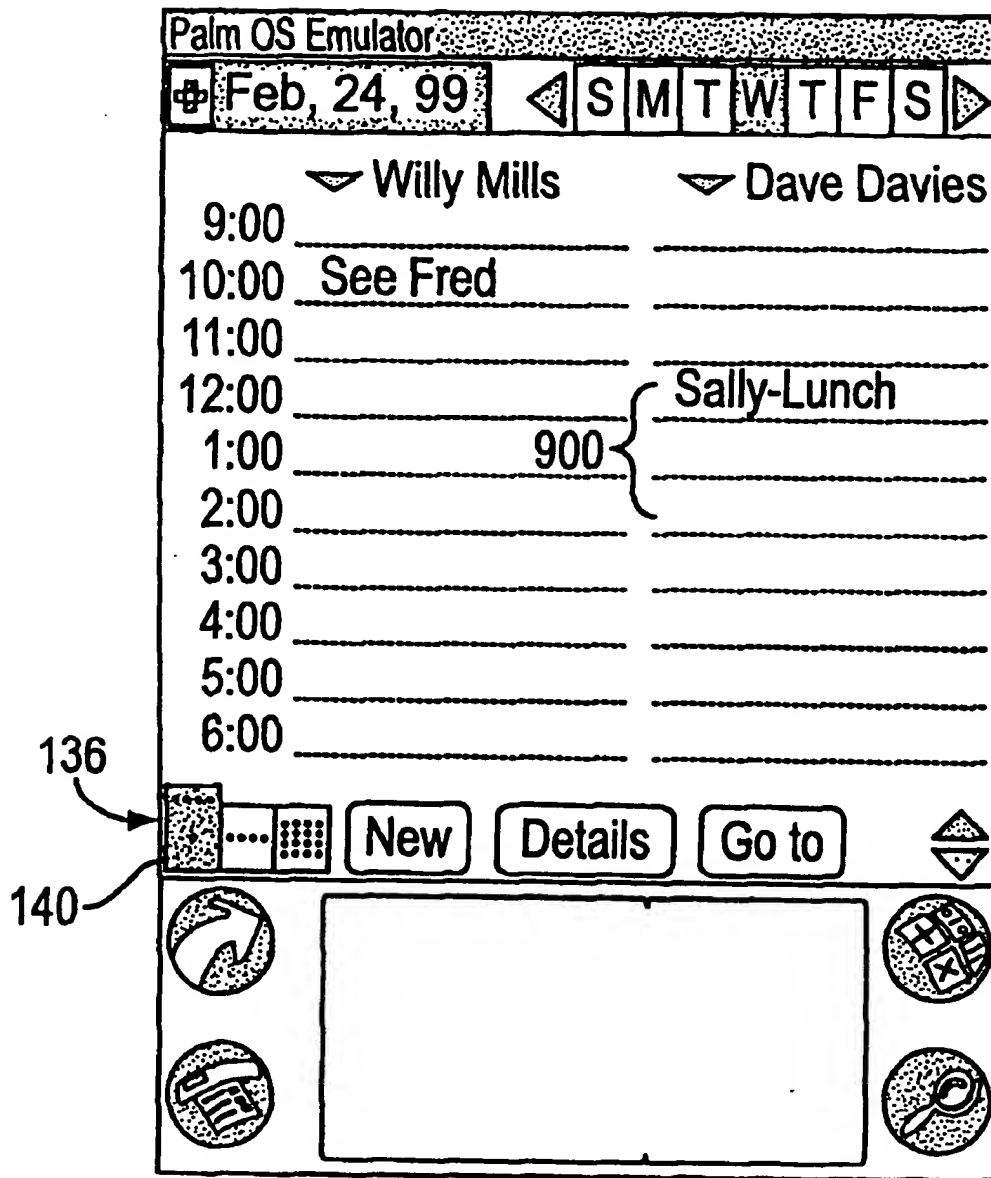


FIG. 9B

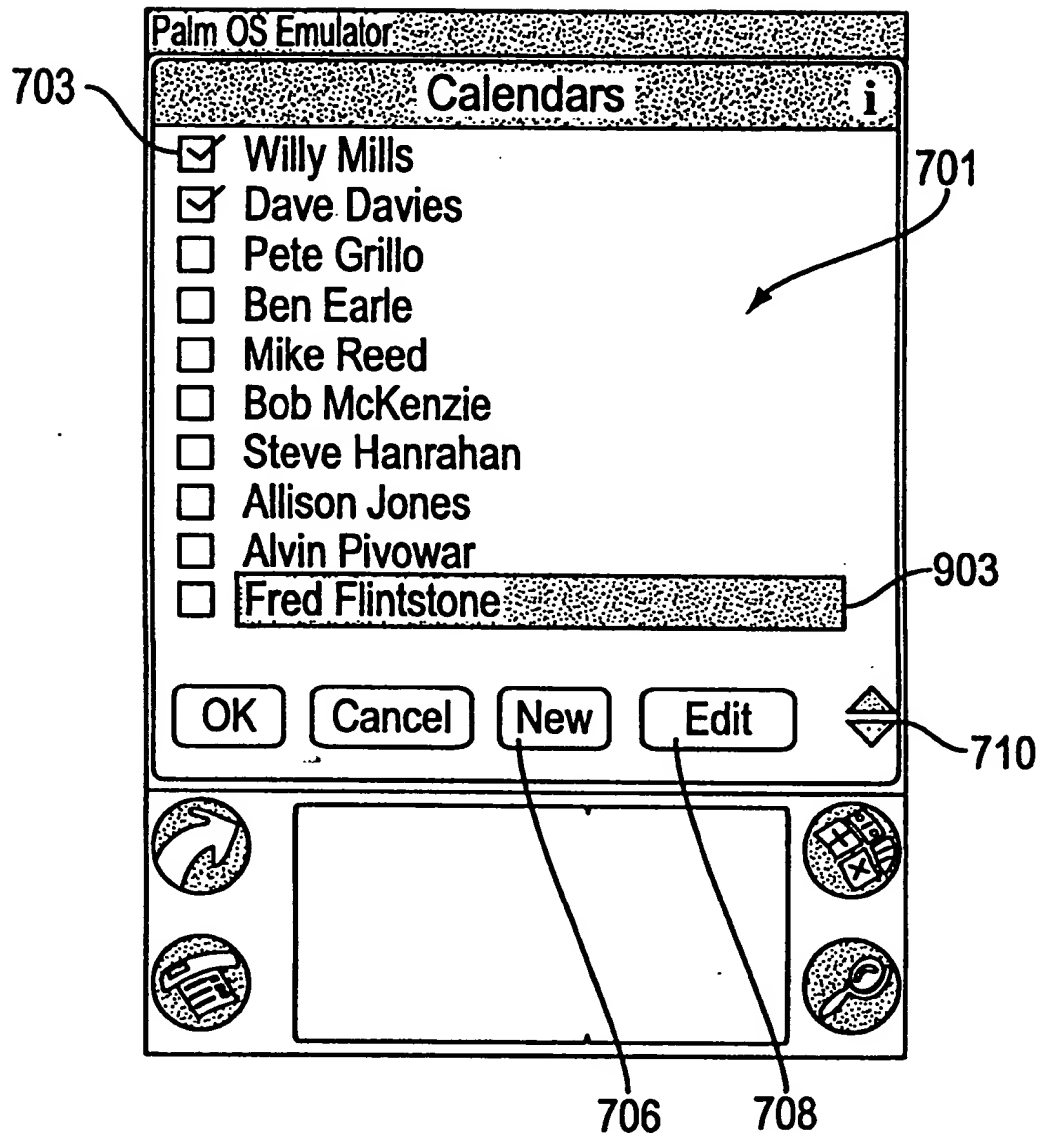


FIG. 9C

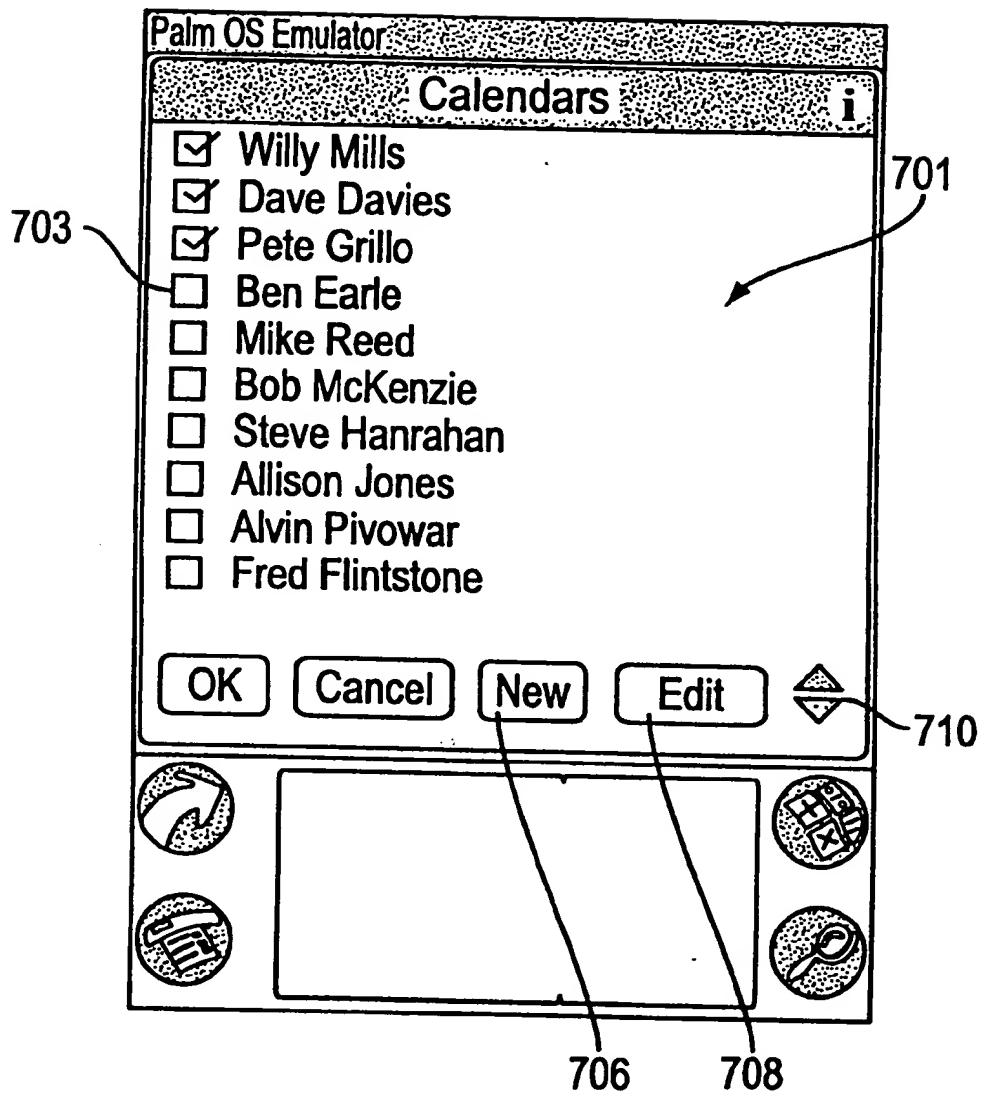


FIG. 9D

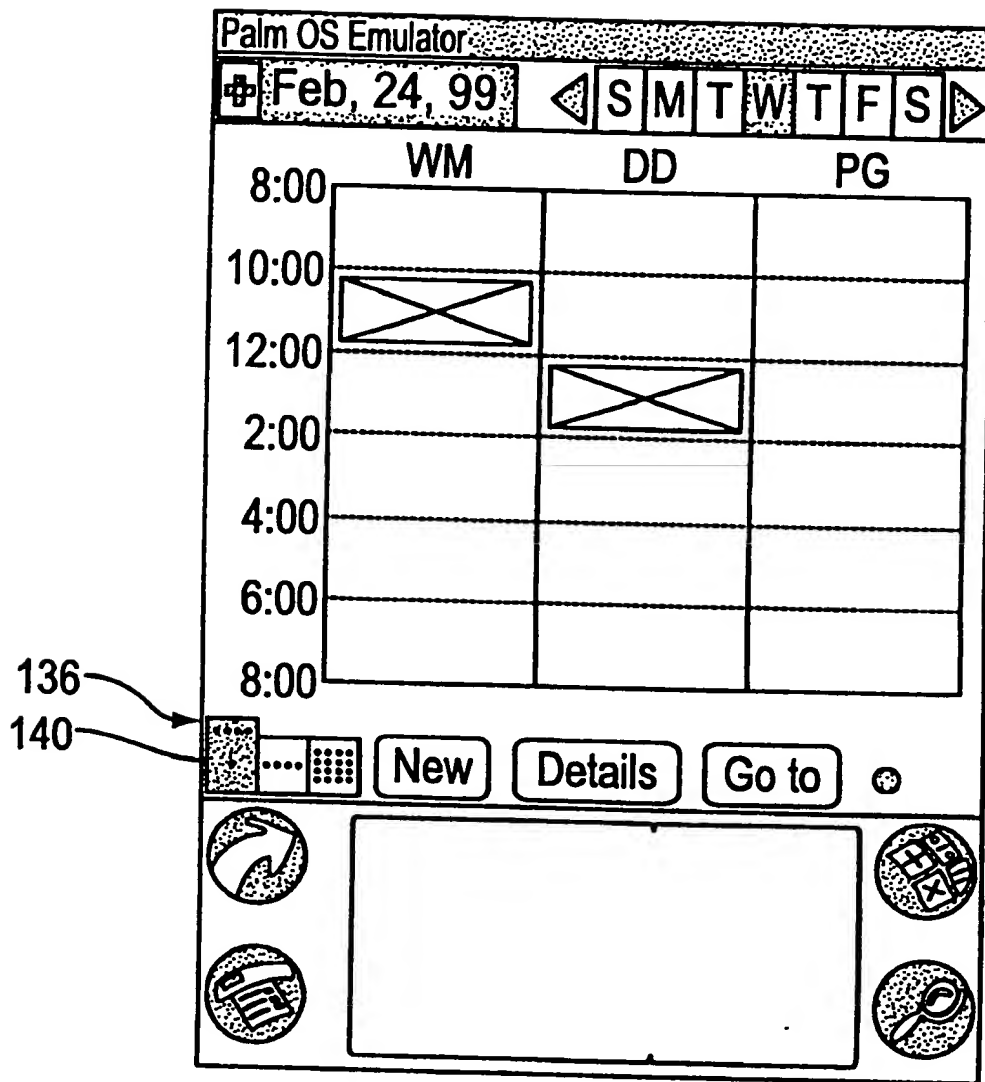


FIG. 9E



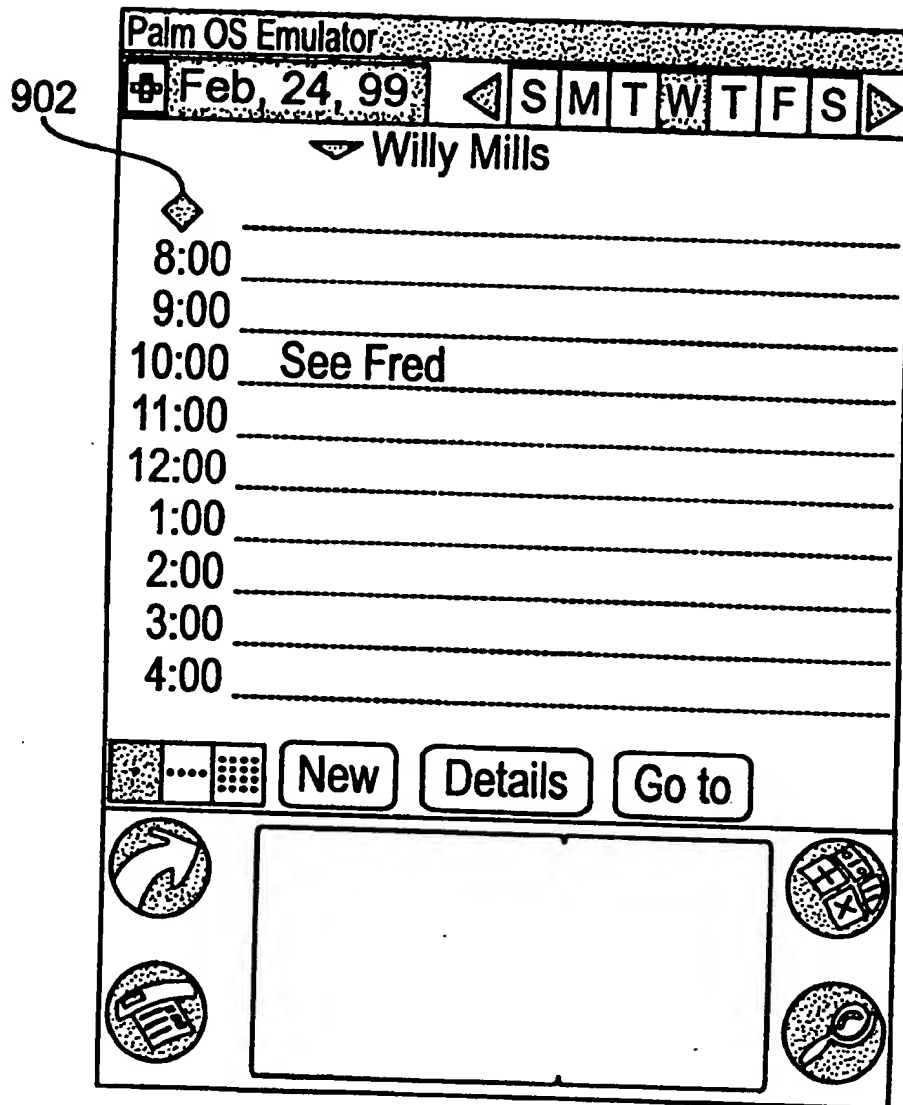


FIG. 9F

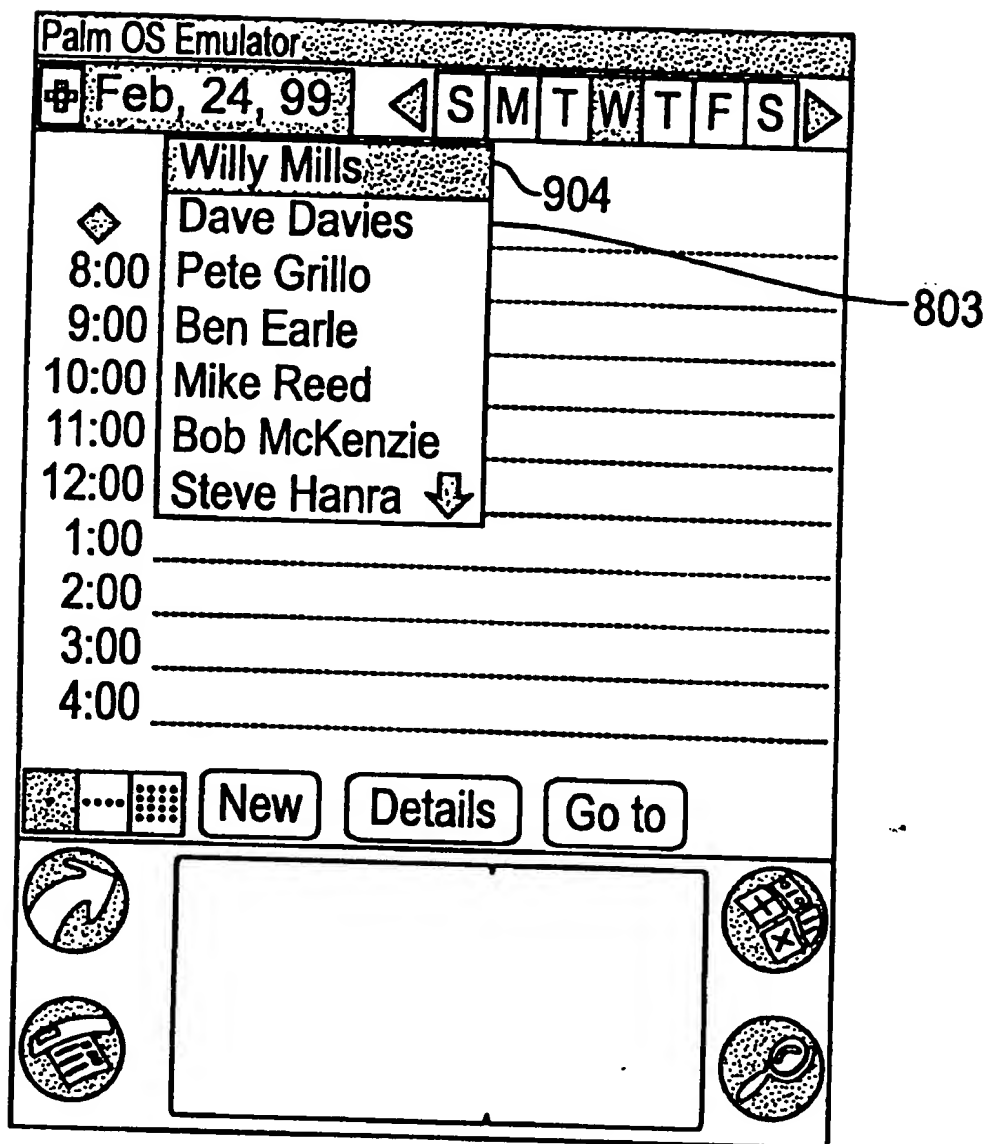


FIG. 9G

1

# SYSTEM AND METHOD FOR DISPLAYING AND MANIPULATING MULTIPLE CALENDARS ON A PERSONAL DIGITAL ASSISTANT

## CROSS REFERENCE TO RELATED APPLICATIONS

The present application is related to co-pending applications entitled "System and Method for Synchronizing Multiple Calendars over a Wide Area Network" by Inventors Alvin Pivowar, Steve Hanrahan and Pete Grillo, Ser. No. 09/289,764, filed concurrently herewith, and incorporated herein by reference; "System and Method for Sharing Data Among a Plurality of Personal Digital Assistants" by Inventors Alvin Pivowar, Steve Hanrahan and Pete Grillo, Ser. No. 09/289,771, filed concurrently herewith, and incorporated herein by reference; "System and Method for Synchronizing Data Among a Plurality of Users Via an Intermittently Accessed Network" by Inventors Alvin Pivowar, Steve Hanrahan and Pete Grillo, Ser. No. 09/289,769, filed concurrently herewith, and incorporated herein by reference; and "System and Method for Advertising during a Data Transfer Process" by Inventors Alvin Pivowar, Steve Hanrahan and Pete Grillo, Ser. No. 09/289,273, filed concurrently herewith, and incorporated herein by reference.

## FIELD OF THE INVENTION

The present invention relates generally to displaying calendars on personal digital assistants and, more particularly, to a system and method for effectively controlling the presentation and manipulation of calendars on a personal digital assistant.

## BACKGROUND OF THE INVENTION

Personal digital assistants, or PDA's, are commonly known hand-held computers that can be used to store, display, and/or manipulate various personal information including, but not limited to contact information, calendar information, etc. Such information can be downloaded from other computer systems, or can be inputted by way of a stylus and pressure sensitive screen of the PDA or any other type of input device such as a mechanical keyboard or a voice recognition module. Examples of PDA's are the Palm™ computer of 3Com Corporation, and Microsoft CE™ computers which are each available from a variety of vendors.

Unlike PDA's, conventional desktop computers, in the past, have allowed the storage and manipulation of multiple calendars thereon. This capability has been prompted by the fact that desktop computers are commonly utilized by multiple users. Further, desktop computers are traditionally equipped with the technical features that are necessary to enable such functionality.

For example, a desktop computer commonly runs software that is capable of allowing various users to share a total capacity of the computer. This may be done by allowing each user to log on and retrieve, add, modify, and delete information, i.e. calendars, that are unique to such user. In terms of technical features, desktops are equipped with an abundance of memory which may be allocated to the calendars of each of the different users. Also, screens of desktop computers are typically have larger than 12" in size. This even allows multiple calendars to be displayed if desired.

Networking of computers has augmented the number of calendars that may be stored on one computer. For instance,

2

a plurality of users may be networked to a single computer, or server, which stores calendars for the users. During operation, each of the users may utilize the server in order to access and manipulate his or her calendar. One example of this can be found on the Internet, wherein various client computers may be connected to the Internet and access one of a plurality of calendars on a single server via a web site.

The Internet has also allowed multiple users to access a single shared group calendar. Such system permits each user to view and edit various scheduled matters on the single calendar. As an option, an electronic message may be sent to each of the users of the group each time the shared group calendar is edited.

In sharp contrast, PDA's currently do not allow the display, let alone the storage of multiple calendars. This is a result of both the limited intended purpose of PDA's and also various technical limitations. For instance, PDA's are traditionally employed for the storage and manipulation of personal data, hence the name personal digital assistant. As such, PDA's conventionally allows the storage of only a single personal calendar.

Even if the storage of multiple calendars on a PDA were desired, many technical obstacles prevent such implementation. This is at least partly due to the portable nature of PDA's which mandates that the various components of PDA's, including the displays, are extremely compact. This feature tends to preclude a feasible method of displaying the multiple calendars in a way that such information may be effectively read and manipulated.

Up to now, the intended purpose and various technical limitations of PDA's has restricted the use of only one calendar per PDA. This has limited PDA users to only organizing his or her own scheduled matters without regard to the scheduled matters of others. Inherent in this limited system is a potential for increased disorganization amongst various PDA owners who interact in normal everyday life.

There is thus a need a system and method for storing multiple calendars on a PDA and further allowing the display of such calendars to enable effective retrieval, addition, modification, and deletion of the calendars.

## DISCLOSURE OF THE INVENTION

A portable, hand-held personal digital assistant (PDA) is provided for simultaneously depicting multiple calendars on a single display. The PDA includes a portable, hand-held housing including a top face, a bottom face, and a side wall therebetween for defining an interior space. An input device is situated on the top face of the housing for allowing input of data. Associated therewith is a display situated on the top face of the housing for depicting data. Situated in the interior space of the housing is memory for storing a plurality of calendars each including a plurality of scheduled matters. Finally, control circuitry is situated in the interior space of the housing and connected between the input device, the display, and the memory. The control circuitry serves for simultaneously depicting a plurality of the calendars on the display. The controller is further adapted for executing multiple methods to facilitate the simultaneous display of the calendars on the display of the PDA. By conveniently displaying the multiple calendars, the present invention allows a user to more effectively manipulate the same.

In order to allow the storage, display, and manipulation of the calendars, the calendars and scheduled matters may be stored in separate calendar databases. Further included is a common database including a plurality of identification data sets each corresponding to the calendar of one of the

3

calendar databases. Such identification data sets each include attributes corresponding to the associated calendar database. Examples of such attributes may indicate that one of the calendars is selected, a primary calendar, read only, and/or a foreign calendar. In operation, the calendars of the calendar databases in accordance with the attributes that are stored in the common database.

Further, various methods may be employed to display the calendars to allow more effective manipulation. For example, in one embodiment of the present invention, at least one calendar is depicted along with a plurality of icons each corresponding to increments of time, i.e. hours, days, and weeks. Next, the present invention allows the selection of one of the icons after which the calendar is divided into increments of time corresponding to the selected icon. As an option, the selected icon is enlarged upon a plurality of calendars being displayed simultaneously.

In another embodiment, upon the selection of a designated icon, a window is depicted which identifies each of the calendars and allows the selection of the calendars by way of any graphical interface such as check boxes. Thereafter, the selected calendars are displayed. While the selected calendars are being displayed, any of the selected calendars may be replaced with another calendar using a pull-down window.

In yet another embodiment, each calendar that is displayed is divided into sections corresponding to increments of time. Further, the scheduled matters are depicted in the sections. In use, a size of the sections is altered as a function of a number of the calendars simultaneously depicted so as to allow a sufficient amount of space for depicting the scheduled matters.

In accordance with still yet another embodiment, a user is allowed to move the scheduled matters of one of the calendars to another one of the calendars. This may be accomplished by dragging the scheduled matter on the display between the calendars.

These and other advantages of the present invention will become apparent upon reading the following detailed description and studying the various figures of the drawings.

#### BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other objects, aspects, and advantages are better understood from the following detailed description of a preferred embodiment of the invention with reference to the drawings, in which:

FIG. 1 is a perspective view of the personal digital assistant of one embodiment of the present invention;

FIG. 2 is a schematic diagram showing the interconnection of the various electrical components of the personal digital assistant of FIG. 1;

FIG. 3A is an illustration of a user interface display of the present invention showing the various features associated therewith;

FIG. 3B is a diagram illustrating a method of displaying multiple calendars on a display the personal digital assistant of FIG. 1 in accordance with one embodiment of the present invention;

FIG. 3C is a block diagram illustrating a data structure in accordance with one embodiment of the present invention;

FIG. 4 is a flowchart illustrating a method of simultaneously displaying multiple calendars on a display of the personal digital assistant of FIG. 1;

FIG. 5 is a flowchart illustrating operation 408 of FIG. 4 in greater detail;

4

FIG. 6 is a flowchart illustrating operation 516 of FIG. 5 in greater detail;

FIG. 7 is a flowchart illustrating operations 410 and 506 of FIGS. 4 and 5, respectively, in greater detail;

FIG. 8 is a flowchart illustrating operation 412 FIG. 4 in greater detail;

FIG. 9A is an illustration of a user interface display of the present invention showing a single calendar in increments of hours;

FIG. 9B is an illustration of a user interface display of the present invention showing a pair of calendars in increments of hours along with a marked duration of a scheduled matter;

FIG. 9C is an illustration of a user interface display of the present invention showing window for selecting which calendars are to be displayed simultaneously;

FIG. 9D is an illustration of a user interface display of the present invention showing window for selecting which calendars are to be displayed simultaneously, wherein an additional calendar is selected by way of a check box;

FIG. 9E is an illustration of a user interface display of the present invention showing three calendars in increments of hours, wherein the sections corresponding to each increment of time is augmented since a large number of calendars are displayed at once;

FIG. 9F is an illustration of a user interface display of the present invention showing one calendar in increments of hours with a marked duration of a scheduled matter along with descriptive text; and

FIG. 9G is an illustration of a user interface display of the present invention showing a pull-down window for selecting one of the calendars to be displayed.

#### DESCRIPTION OF THE PREFERRED EMBODIMENTS

With reference to FIG. 1, a preferred embodiment of the present invention includes a personal digital assistant (PDA) 100. As shown, the PDA 100 includes portable, hand-held housing 102 having a top face, a bottom face, and a side wall therebetween for defining an interior space. Situated on the top face of the housing 102 is an input device 104 which is adapted for allowing input of data. Associated therewith are a plurality of pushbuttons 108 also for input purposes. A display 110 is situated on the top face of the housing 102 for depicting data. It should be noted that the pushbuttons 108, input device 104, and/or the display 110 may be amalgamated into a single device.

As shown in FIG. 2, memory 112 is typically situated in the interior space of the housing 102. In use, the memory 112 serves for storing a plurality of calendars each including multiple scheduled matters. As shown, the memory 112 may take the form of a DRAM or ROM. Also included is a controller 113 situated in the interior space of the housing 102 and connected between the input device 104, the display 110, and the memory 112 via at least one bus 116. It should be noted that the controller 113 may include a microprocessor and accompanying software stored in the memory 112. Alternatively, the controller 113 may take the form of any hardware and/or software combination that is capable of controlling the various components of the present invention in order to carry out the intended functions.

In one embodiment, the PDA 100 may include a hand-held Palm™ PDA available from 3Com Corporation or a Microsoft CE™ computer. In the alternative, the PDA may take the form of any other type of portable data storage module which is capable storing, editing, and/or synchro-

5

nizing sets of personal data. This may be accomplished by any type of I/O mechanisms including, but not limited to a display 110, a plurality of push buttons, a keyboard, a data port, an electronic writing pad using a stylus 106, a voice recognition unit, and/or any other type of I/O device capable of inputting and/or outputting personal data. It should be noted that any of the foregoing I/O devices may be mechanical in nature or, in the alternative, be incorporated into a "touch-sensitive" display.

During use of the PDA 100 of the present invention, various features are displayed during operation in a "calendar" mode. For example, as shown in FIG. 3A the display 110 generally includes a header 120 including a day selector bar 122 having a plurality of day select icons 124, a current date field 126, and a calendar select icon 128. Below the header 120 is a plurality of data fields 129 each corresponding to specific times which are identified by time identifiers 130. The data fields 129 also have a calendar heading 132 and an associated pull-down icon 134 positioned thereabove for reasons that will become apparent hereinafter.

With continuing reference to FIG. 3A, positioned along a lower portion of the display 110 is a time increment selector bar 136 including three time increment icons 140 each corresponding to a unique time increment. Ideally, the icons of the time increment selector bar 136 include three squares each having a number of indicia elements that corresponds to an associated time increment. It should be noted that when selected, the day select icons 124, time increment icons 140, and time identifiers 130 may be highlighted or otherwise distinguished with respect to the remaining icons and identifiers. Also shown in FIG. 3A is a new button 142, a details button 144, and a goto button 146.

It should be noted that while the calendars are being displayed, a user may utilize any one or more of the I/O devices to create, edit, modify various aspects of the calendar information such as data, security rights, or sharing rights. In various alternate embodiments, the foregoing principles may also be applied to other information such as contact information.

FIG. 3B generally shows the operation of one embodiment of the present invention. In use, the controller of the PDA 100 is adapted for allowing a user to simultaneously depict a plurality of the calendars 146 on the display 110. This is accomplished by permitting operation in a plurality of states. For example, in a first state 300, a single primary, or default, calendar is displayed. A second state 302 is used to depict a single calendar other than the primary calendar. In still yet another state, a third state 304, a plurality of calendars may be depicted.

During use, a user may maneuver between the various states by selecting certain items on the PDA 100. For example, while in the first state 300, a user may shift to the second state by selecting the pull-down icon 134 which in turn provides a pull-down window for allowing the selection of any available calendars in place of the primary calendar in the second state 302. In the alternative, the calendar select icon 128 may be selected which provides a separate full-size window for allowing the selection of multiple calendars to be displayed simultaneously in the third state 304. Further details regarding the operation of the pull-down window and full-size window will be set forth later.

While in the second state 302, the pull-down icon 134 may be used in a manner similar to that in the first state 300 in order to select one of the available calendars in place of the currently displayed calendar. Further, the calendar select icon 128 may be selected to provide the full-size window for

6

allowing the selection of multiple calendars to be displayed simultaneously in the third state 304.

In the third state 304, the pull-down icon 134 of any of the calendar headings 132 can be used in a manner similar to that in the previous states in order to select one of the available calendars in place of the currently displayed calendars. Further, the calendar headings 132 may be selected for reverting to the second state 302, herein a single calendar is displayed.

As shown in FIG. 3B, in each of the states, the calendars may be divided into various selected increments of hours, days, and weeks. It should be noted that the calendars may include any type of calendars including a sports calendar, a personal calendar, a work-related calendar, and/or another person's calendar. Such calendars may be manually inputted, downloaded, or synchronized in any fashion.

FIG. 3C is a block diagram illustrating a data structure which facilitates the display of multiple calendars on a display 110 of the PDA 100 of FIG. 1. In order to facilitate handling the various calendars stored within the PDA 100, each of the calendars and associated scheduled matters may be stored in separate databases 150. Further, a common database 152 may be provided including a plurality of identification data sets each corresponding to the calendar of one of the calendar databases.

In one embodiment, each identification data set includes a system name, i.e. CAL0, CAL1, CAL2, etc.; a username, i.e. Willie Mills, Dave Davies, etc.; and a plurality of attributes. As shown in FIG. 3C, such attributes may indicate that one of the calendars is selected, a primary calendar (default), read only, or a foreign calendar. It should be noted that the attributes may be selectively determined by the user or automatically assigned depending on a source of the associated calendar. In use, the common database may be referenced to display the calendars of the calendar databases in accordance with the attributes. Further, the common database allows the scheduled matters to be shared among the calendar databases.

It should be noted that the data structure of FIG. 3C is further critical for allowing the features of the present invention to be utilized with PDA's that are capable of handling only a single calendar. This backwards compatibility is enabled by allowing the data of each of the calendars including the original calendar to be stored independently. The correlation data in the form of attributes, on the other hand, is stored in a separate common database.

FIG. 4 shows a more detailed flowchart of the method of simultaneously depicting a plurality of the calendars on the display. In operation 400 of FIG. 4, the primary calendar, as indicated by the attributes, is displayed, as shown in FIG. 9A. Next, in decisions 402 and 404, a wait loop is executed until a multiple-view event, i.e. an event that requires the display of multiple calendars, is detected. If such event is not detected, the display continues normally in operation 406 of FIG. 4. Note FIG. 9A.

If, however, a multiple-view event is detected, it is then determined which type of multiple-view event has taken place. It should be noted that a multiple-view event may include the selection of the calendar select icon 128, one of the time increment icons 140, or the pull-down icon 134. If it is determined that one of the time increment icons 140 was selected in decision 404, the multiple views are handled in operation 408. If, on the other hand, it is determined that the calendar select icon 128 was selected in decision 404, the selected calendars are handled in operation 410. Finally, if it is determined that the pull-down icon 134 was selected in

7

decision 404, the calendars are picked in operation 412. Additional details regarding operations 48, 410, and 412 will be set forth hereinafter with reference to FIGS. 5, 7, and 8.

FIG. 5 is a more detailed flowchart delineating the method associated with the operation 408 shown in FIG. 4 when one of the time increment icons 140 is selected. As shown in FIG. 5, it is first determined which calendars are active, or selected, in operation 500 after which such active calendars are displayed side-by-side in operation 502 and divided into the time increments associated with the time increment icon that was selected. In other words, the calendar(s) is divided into increments of time corresponding to one of the time increment icons 140 that is currently selected. As an option, the selected icon may be altered, i.e. enlarged, upon a plurality of calendars being displayed simultaneously, wherein the selected time increment icon is augmented as a function of a number of the calendars being displayed simultaneously. Note, for example, FIGS. 9B and 9E.

With continuing reference to FIG. 5, after the selected calendars are displayed, it is then determined in decision 504 whether the calendar select icon 128 has been selected. If so, the selected calendars are handled in operation 506. If not, it is then determined in decision 508 whether a specific time, i.e. date, has been selected. If so, then a portion of the process is repeated. If a specific time is not determined in decision 508, it is determined in decision 510 whether one of the time increment icons 140 has been selected. If so, the present method ceases. If not, however, it is then determined whether a calendar heading 132 or an event, i.e. scheduled matter, has been selected in decisions 512 and 514, respectively. Thereafter, the calendar heading 132 or event is handled in operations 516 and 518, respectively.

FIG. 6 is a more detailed flowchart illustrating the method associated with operation 516 shown in FIG. 5. In particular, as shown in decision 600 of FIG. 6, it is first determined whether there is more than one calendar displayed on the PDA 100 or, in other words, whether the present invention is operating in the third state 304 of FIG. 3B. If the present invention is operating in the third state 304, details relating to the instant event, or scheduled matter, are presented in an unillustrated pop-up window in operation 602. At that point, the user may decide whether to move the instant event in decision 604. If so, in operation 606, the event may be moved to another one of the simultaneously displayed calendars by dragging the scheduled matter on the display between the calendars using the stylus 106 or any other input device.

With continuing reference to FIG. 6, it is shown that the events, or scheduled matters, of the calendars may be modified. This is accomplished by first determining whether the detail button 144 has been selected in decision 608. If it has, the event is displayed in operation 610 for modification if desired in operation 612.

FIG. 7 is a more detailed flowchart delineating the method associated with operations 410 and 506 shown in FIGS. 4 and 5, respectively, when the calendar select icon 128 is selected. Upon such selection, a window 701 is displayed which identifies each of the calendars. Note FIGS. 9C and 9D. As indicated in decision 700 and operation 702, a user may select among the calendars by toggling through the identifiers and using the check boxes 703 of the window. Thereafter, the calendars may be edited, added, or moved in operations 704 using a new button 706, edit button 708, and a pair of arrow buttons 710.

Finally, FIG. 8 is a more detailed flowchart delineating the method associated with operation 412 shown in FIG. 4 when

8

the pull-down icon 134 was selected. As shown, it is first determined whether the calendar associated with the pull-down icon 134 is a primary calendar in decision 800. If so, then any selected calendar is shown in addition to the primary calendar in operation 802. If, however, the calendar associated with the pull-down icon 134 is not a primary calendar and there are not many calendars that are selected (1 or 2), then a pull-down window 803 is displayed in operation 804. Note FIG. 9G. Thereafter, a new calendar may be selected in operation 806 after which such selected calendar replaces the previous calendar in operation 808 in the corresponding section of the display. Finally, the pull-down window is disabled in operation 810.

With specific reference now to FIGS. 9A-9G, various graphical user interfaces are shown that may occur during use of the present invention. FIG. 9A depicts a single calendar divided into increments of hours. As shown, the time increment icons 140 of the time increment selector bar 136 are of a similar size when a single calendar is displayed.

FIG. 9B shows a pair of calendars displayed simultaneously in a side-by-side relationship and each divided into increments of hours. It should be noted that the time increment icon 140 that corresponds to the increments of hours is enlarged since multiple calendars are displayed. Further, a time duration bar 900 is included for indicating a time period during which a scheduled matter is arranged.

FIG. 9C depicts the full-size window 701 which displays all of the calendars available to be picked. As shown, checkboxes 703 are available to facilitate such selection. As mentioned earlier, in order for the full-size window 701 to be displayed, the calendar select icon 128 of FIG. 3A must be selected. FIG. 9D also shows the full-size window 701, but with an additional selected calendar. As shown, selection of a calendar is facilitated by way of a highlight bar 903.

FIG. 9E depicts three calendars displayed simultaneously in a side-by-side relationship and each divided into increments of hours. As shown, the sections of each calendar are enlarged to compensate for the smaller areas in which the calendars are fitted. In the present display, the time increment icon 140 that corresponds to the increments of hours is enlarged.

FIG. 9F shows a single calendar similar to that of 9A with the exception of an open appointment icon 902 that indicates that a specific time period is open. FIG. 9G is an illustration showing the pull-down window 803 which may be accessed by selecting the pull-down icon 134. In one embodiment, the pull-down window requires only a part of the display 110 of the PDA 100. As shown, a currently selected calendar is indicated by way of a highlight bar 904.

While various embodiments have been described above, it should be understood that they have been presented by way of example only, and not limitation. Thus, the breadth and scope of a preferred embodiment should not be limited by any of the above described exemplary embodiments, but should be defined only in accordance with the following claims and their equivalents.

What is claimed is:

1. A portable data storage module for simultaneously depicting multiple calendars on a single display comprising:
  - a portable, hand-held housing including a top face, a bottom face, and a side wall therebetween for defining an interior space;
  - an input device situated on the top face of the housing and adapted for allowing input of data;
  - a display situated on the top face of the housing and adapted for depicting data;

9

memory situated in the interior space of the housing for storing a plurality of calendars each including a plurality of scheduled matters; and

a controller situated in the interior space of the housing and connected between the input device, the display, and the memory, the controller suitable for simultaneously depicting a plurality of the calendars on the display.

2. The portable data storage module as recited in claim 1, wherein the scheduled matters are depicted on the display with each calendar.

3. The portable data storage module as recited in claim 1, wherein the calendars are divided into increments of hours.

4. The portable data storage module as recited in claim 1, wherein the calendars are divided into increments of days.

5. The portable data storage module as recited in claim 1, wherein the calendars are divided into increments of weeks.

6. The portable data storage module as recited in claim 1, wherein the controller is suitable for manipulating the calendars.

7. A method for controlling the presentation of at least one calendar on a display of a portable data storage module comprising the operations of:

storing various calendars within a portable data storage module in separate databases,

depicting at least one calendar on a display of a portable data storage module, the display situated on a top face of the portable data storage module;

depicting a plurality of icons each corresponding to increments of time selected from the group of increments of time including hours, days, and weeks;

allowing the selection of one of the icons; and

dividing the at least one calendar into increments of time corresponding to one of the icons that is selected.

8. The method as recited in claim 7, and further comprising the operation of:

altering one of the icons upon a plurality of calendars being displayed simultaneously.

9. The method as recited in claim 8, wherein the selected icon is altered upon a plurality of the calendars being displayed simultaneously.

10. The method as recited in claim 9, wherein the selected icon is altered as a function of a number of the calendars being displayed simultaneously.

11. A method for controlling the presentation of a plurality of calendars on a display of a portable data storage module comprising the operations of:

storing various calendars within a portable data storage module in separate databases;

providing a window on a display of a portable data storage module which identifies each of the calendars, the display situated on a top face of the portable data storage module;

allowing the selection of the identified calendars displayed in the window; and

simultaneously displaying all of the selected calendars.

12. The method as recited in claim 11, wherein upon a plurality of calendars being selected, each of the selected calendars are depicted simultaneously.

13. The method as recited in claim 12, wherein upon a plurality of calendars being selected, one of the selected calendars may be replaced with another calendar.

14. The method as recited in claim 11, wherein the selection of the calendars is executed using check boxes.

10

15. The method as recited in claim 11, wherein the window is enabled upon selecting an icon.

16. The method as recited in claim 11, wherein the window is a pull-down window.

17. The method as recited in claim 11, wherein each calendar that is selected is given a calendar heading.

18. A method for controlling the presentation of a plurality of calendars on a display of a portable data storage module comprising the operations of:

storing various calendars within a portable data storage module in separate databases;

depicting a plurality of calendars simultaneously on a display of a portable data storage module, the display situated on a top face of the portable data storage module, wherein each calendar is divided into sections corresponding to increments of time and scheduled matters are depicted in the sections; and

altering a size of the sections as a function of a number of the calendars simultaneously depicted.

19. The method as recited in claim 18, wherein the size of the sections is inversely proportional to the number of calendars simultaneously depicted.

20. A method for controlling the presentation of a plurality of calendars on a display of a portable data storage module comprising the operations of:

storing various calendars within a portable data storage module in separate databases;

depicting a plurality of calendars with scheduled matters on a display of a portable data storage module, the display situated on a top face of the portable data storage module; and

allowing movement of the scheduled matter of one of the calendars to another one of the calendars.

21. The method as recited in claim 20, wherein scheduled matter is moved by dragging the scheduled matter on the display between the calendars.

22. A method for simultaneously depicting multiple calendars on a display of a portable data storage module comprising the operations of:

providing a plurality of calendar databases each including a calendar having a plurality of scheduled matters;

providing a common database including a plurality of identification data sets each corresponding to the calendar of one of the calendar databases, the identification data sets each including attributes corresponding to the calendar database; and

displaying the calendars of the calendar databases on a top face of the portable data storage module in accordance with the attributes.

23. The method as recited in claim 22, wherein one of the attributes indicates that one of the calendars is selected.

24. The method as recited in claim 22, wherein one of the attributes indicates that one of the calendars is a primary calendar.

25. The method as recited in claim 22, wherein one of the attributes indicates that one of the calendars is read only.

26. The method as recited in claim 22, wherein one of the attributes indicates that one of the calendars is a foreign calendar.

27. The method as recited in claim 22, and further comprising the operation of:

manipulating the calendars of the calendar databases.

\* \* \* \* \*



US006369840B1

(12) **United States Patent**  
Barnett et al.

(10) Patent No.: **US 6,369,840 B1**  
(45) Date of Patent: **Apr. 9, 2002**

(54) **MULTI-LAYERED ONLINE CALENDARING AND PURCHASING**

(56) **References Cited**

**U.S. PATENT DOCUMENTS**

(75) Inventors: **Theodore H. Barnett; Anthony A. Espinoza; Vengpui Louis Lao**, all of San Francisco; **David C. Sobotka**, Redwood City; **Andrew W. Zaeske**, Los Altos, all of CA (US)

5,596,373 A \* 1/1997 White et al. .... 348/569  
5,960,406 A \* 9/1999 Rasansky et al. .... 705/9  
6,018,343 A \* 1/2000 Wang et al. .... 345/356  
6,064,977 A \* 5/2000 Haverstock et al. .... 705/9  
6,085,235 A \* 7/2000 Clarke, Jr. et al. .... 709/219

(73) Assignee: **America Online, Inc.**, Dulles, VA (US)

\* cited by examiner

(\*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

*Primary Examiner*—John Cabeca

*Assistant Examiner*—Kieu D. Vu

(74) *Attorney, Agent, or Firm*—Michael A. Glenn

(57) **ABSTRACT**

(21) Appl. No.: **09/265,515**

(22) Filed: **Mar. 10, 1999**

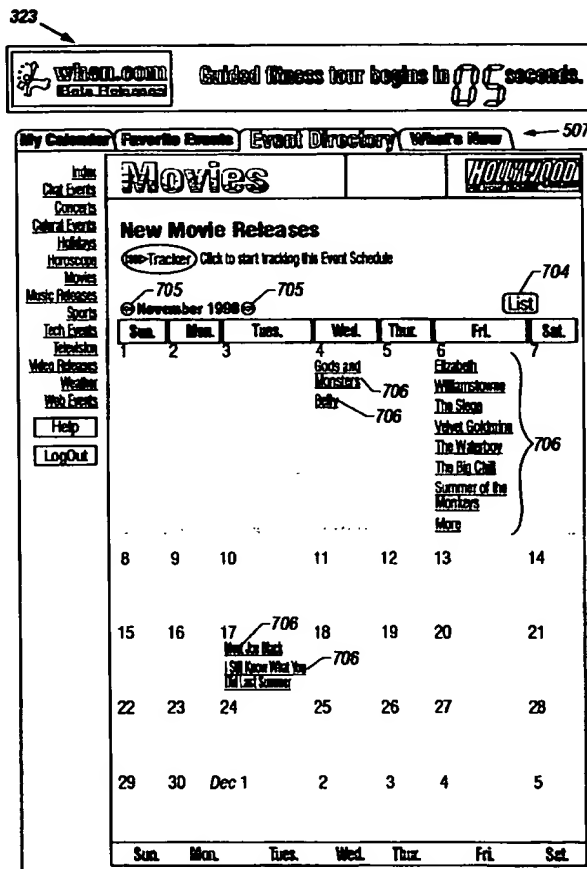
(51) Int. Cl.<sup>7</sup> ..... **G06F 3/00**

(52) U.S. Cl. .... **345/853; 345/751; 345/963; 345/962; 345/841; 345/854**

(58) **Field of Search** ..... **345/113, 326, 345/329, 339, 344, 345, 963, 435, 353, 356, 357, 331, 853, 733, 962, 751, 841, 854, 764, 797; 709/201, 203, 219, 217; 707/501**

A computer-implemented method and system for generating and displaying a calendar containing user-selected events from user-selected categories. A plurality of categories of events are provided. The user can select which categories are of interest, and then select individual events within those categories. Events are overlaid on a calendar unique to the user. Calendars may also be shared among a number of selected users, if desired. Online purchasing and related actions can be associated with each event.

**56 Claims, 24 Drawing Sheets**





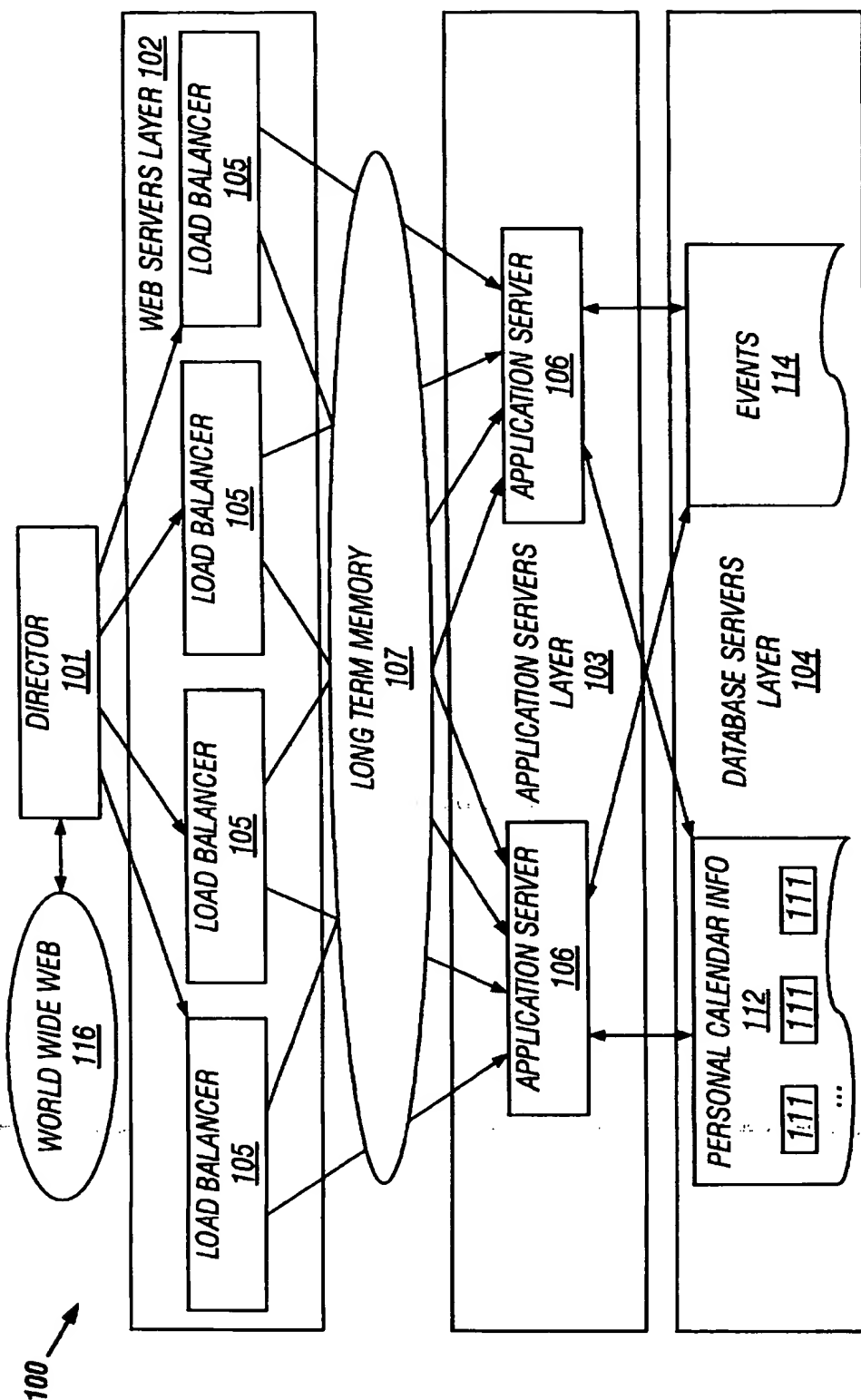


FIG. 1

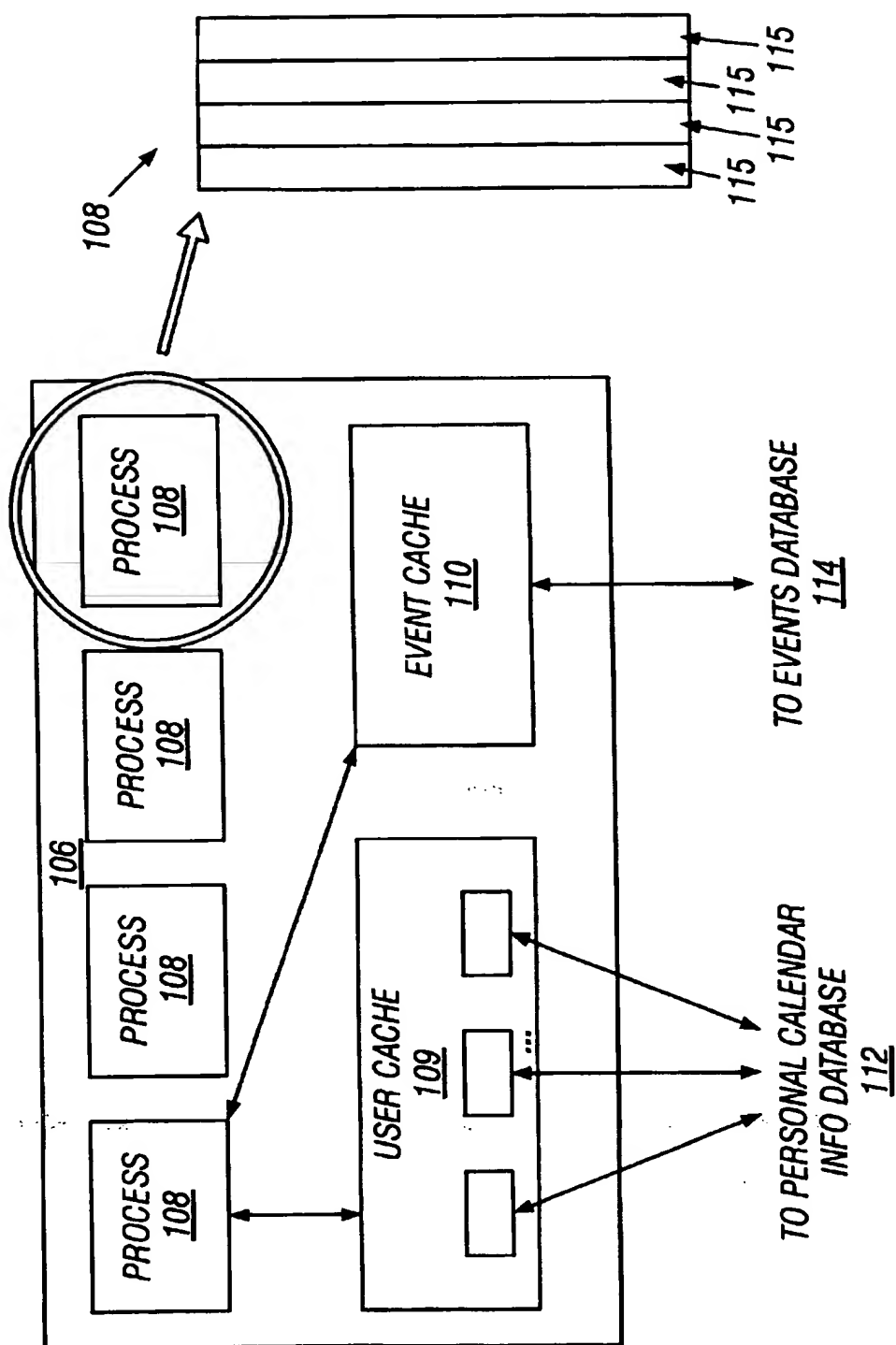


FIG. 1A

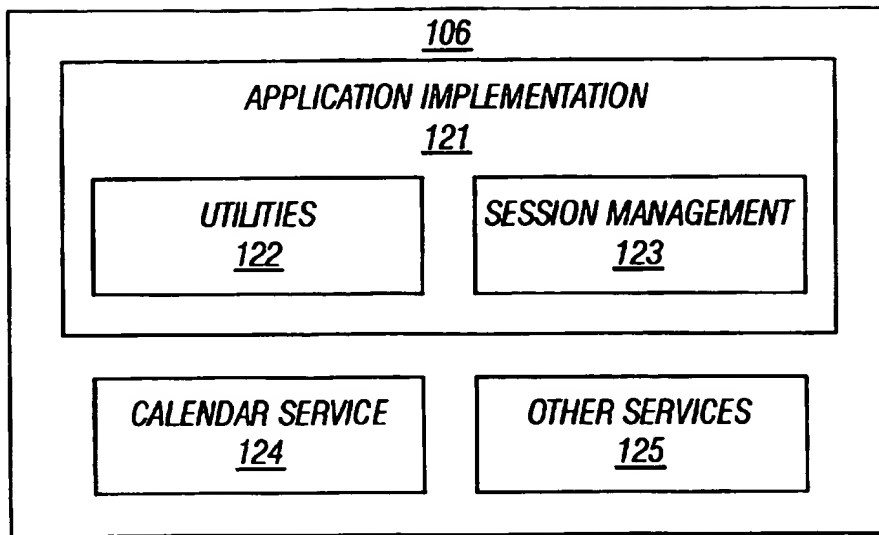


FIG. 1B

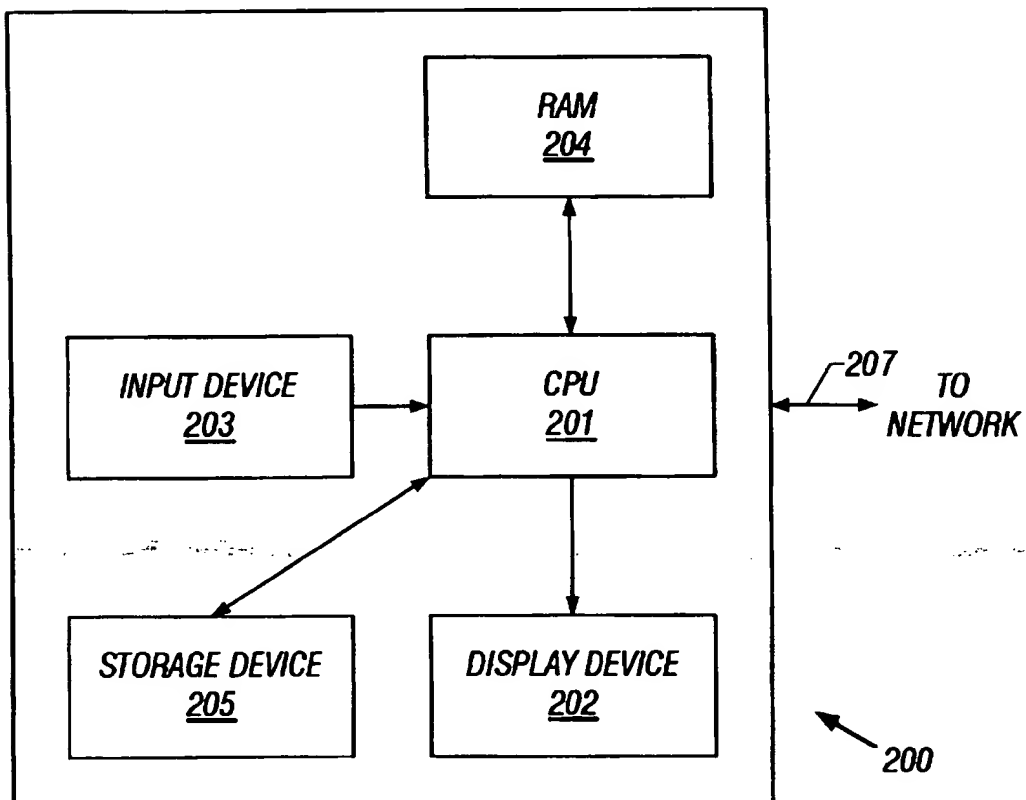


FIG. 2

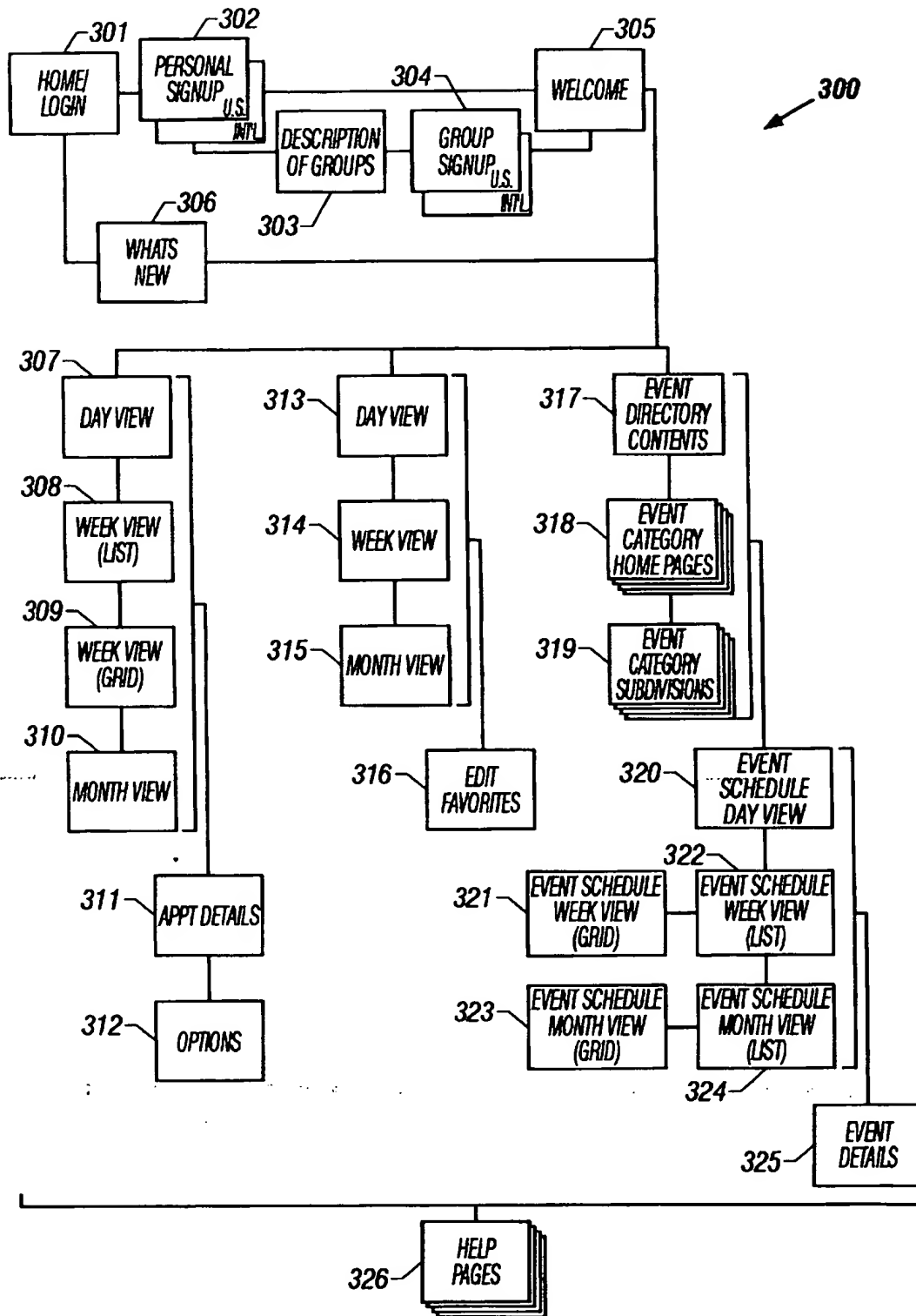


FIG. 3

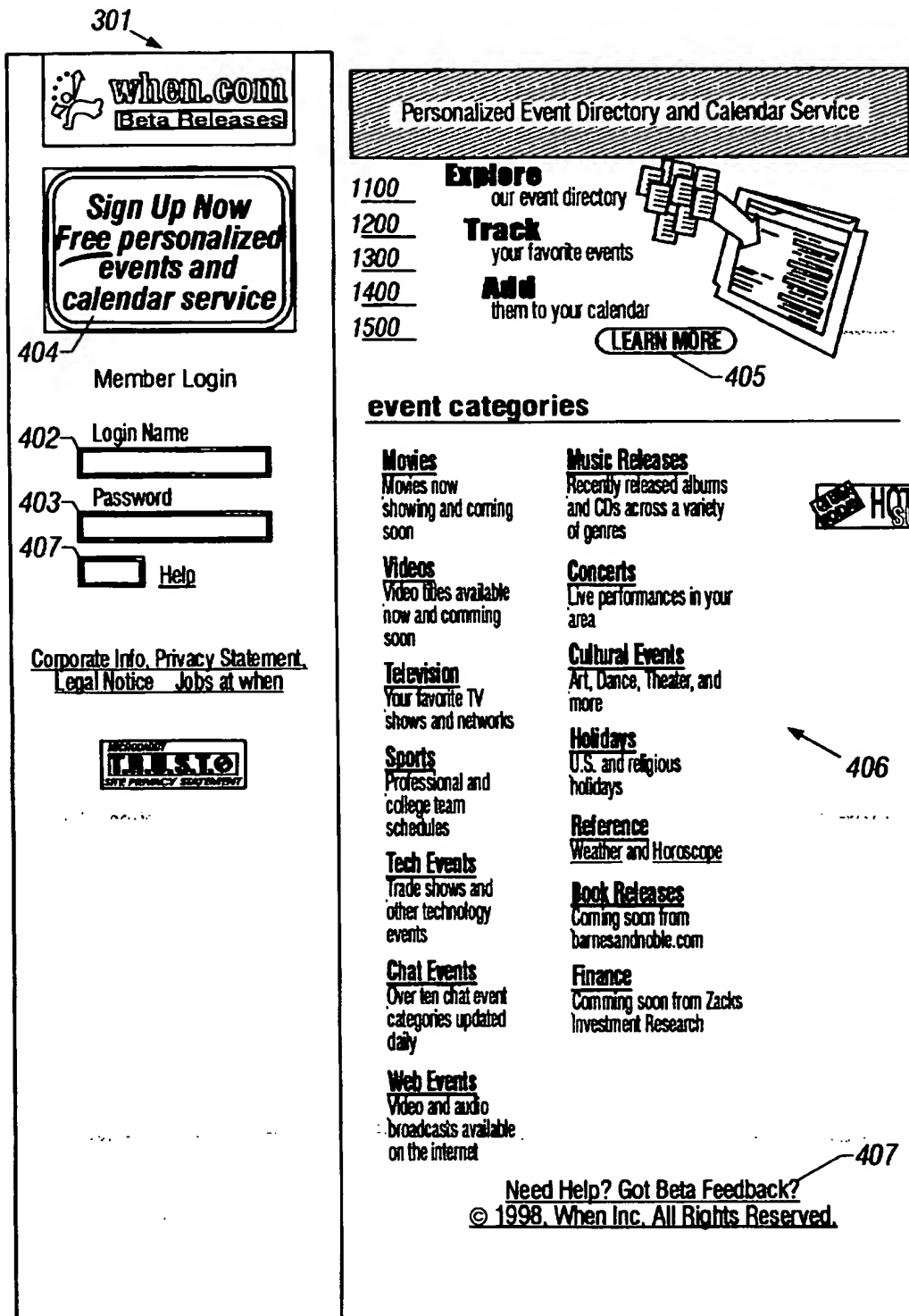
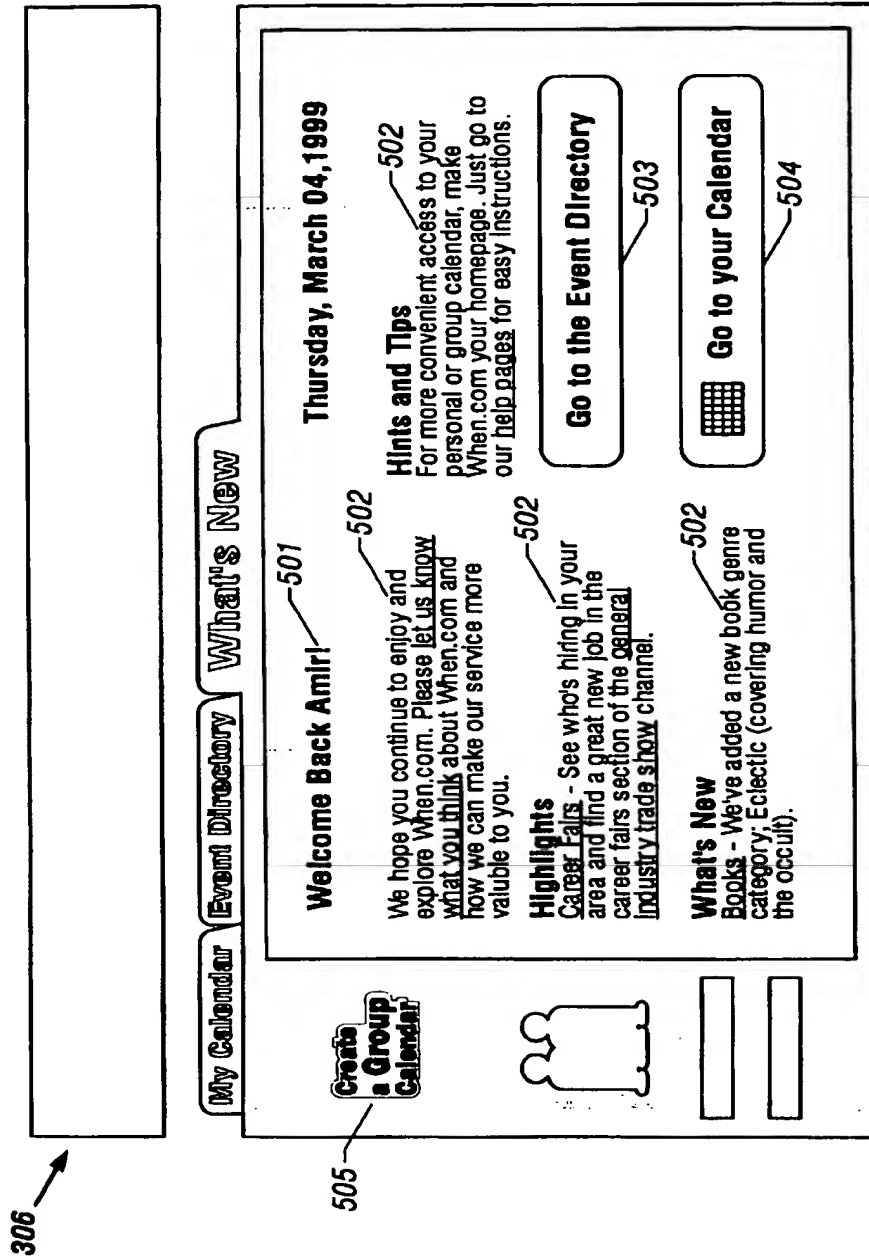


FIG. 4



Corporate Info | Jobs at When | Privacy Policy  
 Feedback & Email Support | Help  
 © 1999, When Inc. All Rights Reserved

FIG. 5

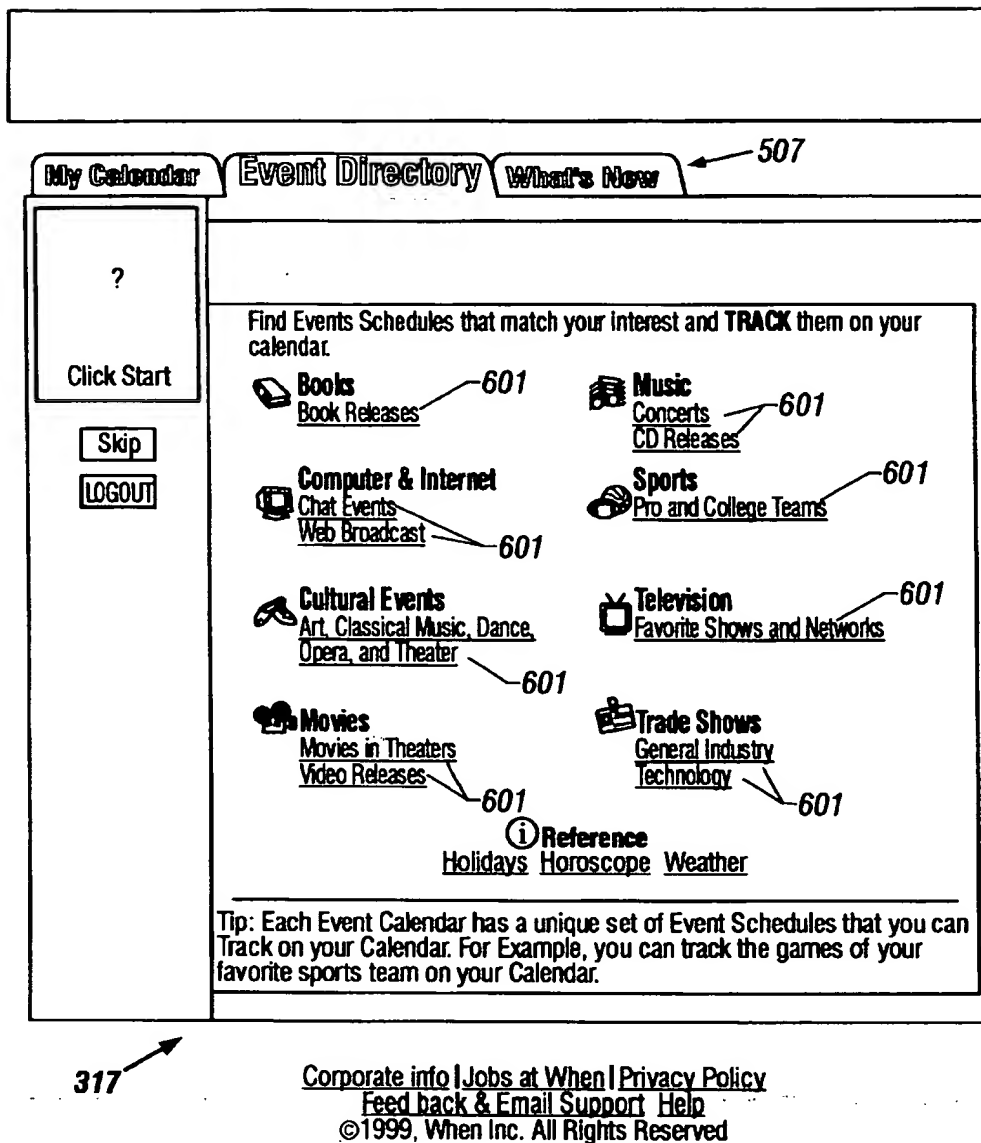



FIG. 6

324



If we don't have your book, nobody does...

My Calendar
Favorite Events
Event Directory
What's New


- [Index](#)
- [Chat Events](#)
- [Concerts](#)
- [Cultural Events](#)
- [Holidays](#)
- [Horoscope](#)
- [Movies](#)
- [Music Releases](#)
- [Sports](#)
- [Tech Events](#)
- [Television](#)
- [Video Releases](#)
- [Weather](#)
- [Web Events](#)

Help

LogOut

## Movies

**New Movie Releases**

 **Tracker** Click to start tracking this Event Schedule

705 702 705

⊖ November 1998 ⊕

Grid 703

DATE & TIME	TITLE	COMMENTS
Wed, Nov 04, All Day	<u>Gods and Monsters</u>	Drama, NR
Wed, Nov 04, All Day	<u>Belly</u>	Drama, R
Fri, Nov 06, All Day	<u>Elizabeth</u>	Drama, R
Fri, Nov 06, All Day	<u>Williamstowne</u>	Drama, NR
Fri, Nov 06, All Day	<u>The Siege</u>	Action/Adventure, NR
Fri, Nov 06, All Day	<u>Velvet Goldmine</u>	Drama, R
Fri, Nov 06, All Day	<u>The Waterboy</u>	Comedy, PG-13
Fri, Nov 06, All Day	<u>The Big Chill</u>	Drama, R
Fri, Nov 06, All Day	<u>Summer of the Monkeys</u>	Family, G
Fri, Nov 06, All Day	<u>The Wizard of Oz</u>	Family, G
Tue, Nov 17, All Day	<u>Meet Joe Black</u>	Drama, PG-13
Tue, Nov 17, All Day	<u>I Still Know What You Did Last Summer</u>	Horror, R

⊖ November 1998 ⊕


705 705

Grid 703

FIG. 7A



323



Guided fitness tour begins in **05** seconds.

My Calendar Favorite Events Event Directory What's Now 507

Index

Chat Events

Concerts

Cultural Events

Holidays

Horoscope

Movies

Music Releases

Sports

Tech Events


Television

Video Releases


Weather

Web Events

## Movies



### New Movie Releases

 **Tracker** Click to start tracking this Event Schedule

705
705
704

November 1998

Sun.	Mon.	Tues.	Wed.	Thur.	Fri.	Sat.
1	2	3	4	5	6	7
			<u>Gods and Monsters</u> <u>Belly</u>		<u>Elizabeth</u> <u>Williamstowne</u> <u>The Siege</u> <u>Velvet Goldmine</u> <u>The Waterboy</u> <u>The Big Chill</u> <u>Summer of the Monkeys</u> <u>More</u>	
8	9	10	11	12	13	14
15	16	17	18	19	20	21
		<u>Meet Joe Black</u> <u>I Still Know What You Did Last Summer</u>				
22	23	24	25	26	27	28
29	30	Dec 1	2	3	4	5
Sun.	Mon.	Tues.	Wed.	Thur.	Fri.	Sat.

FIG. 7B

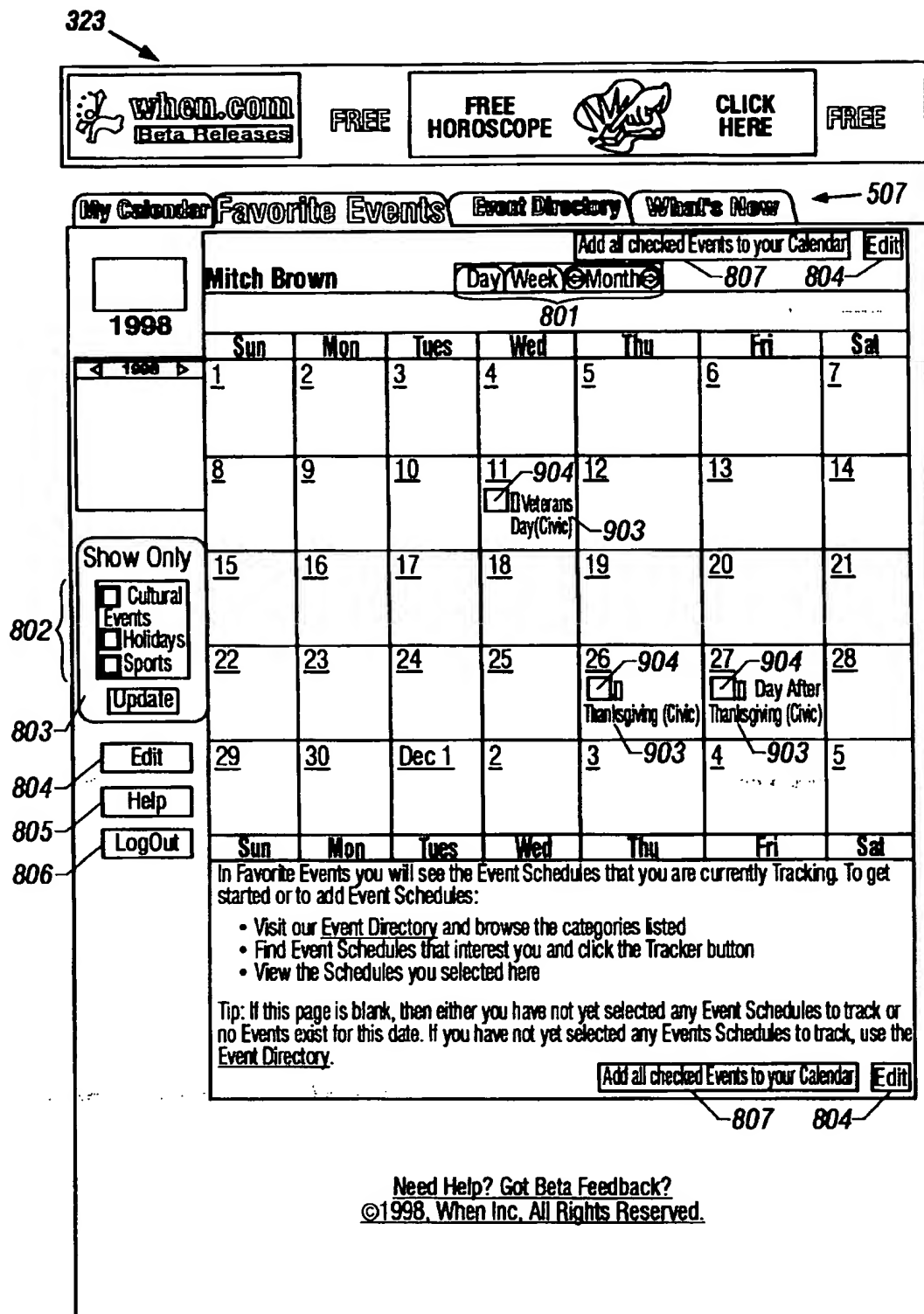


FIG. 8

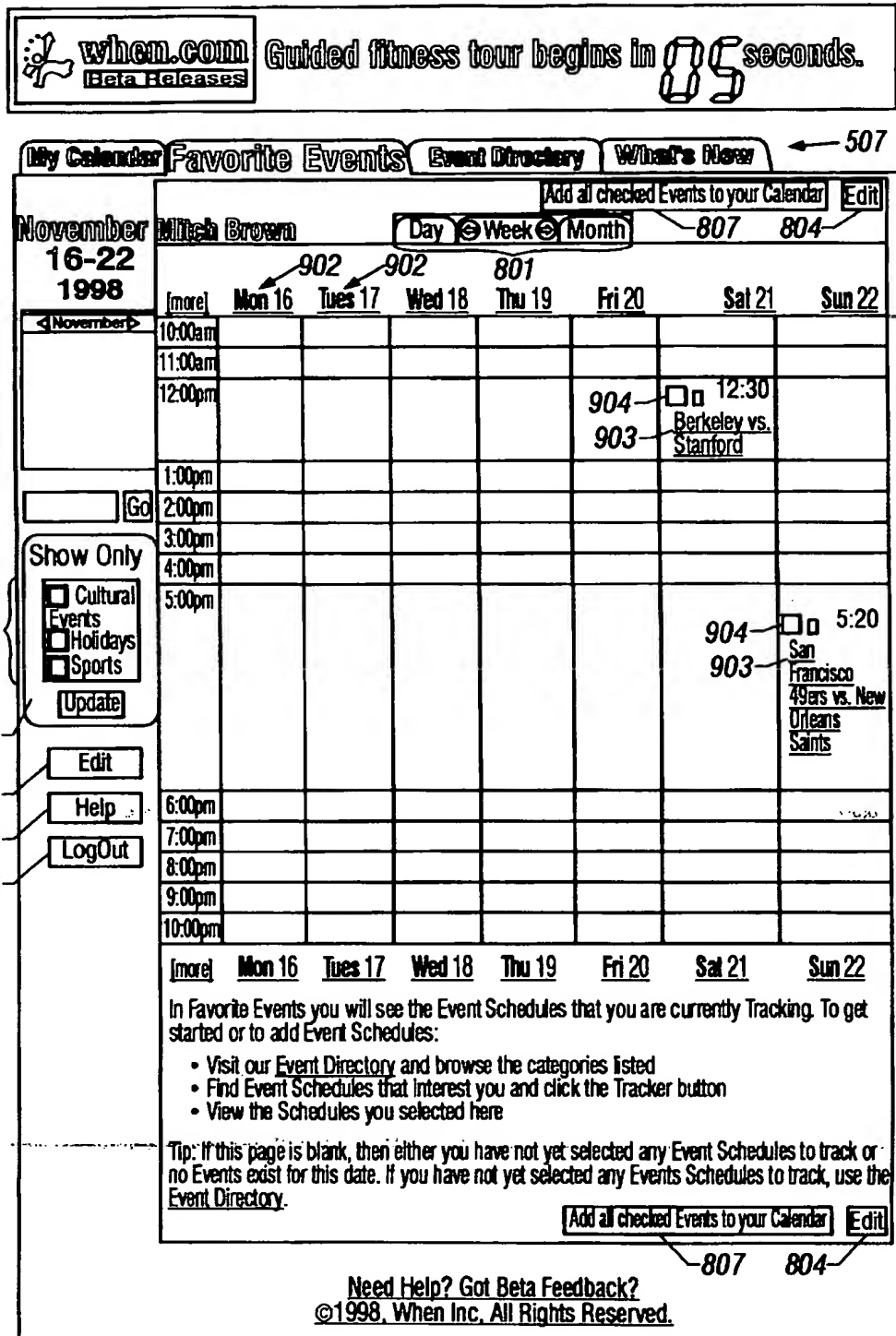



FIG. 9



We'll, the least you can do is let us  
give you the forecast.

My Calendar

**Tuesday**  
11/17/1998

< November >

Go

Edit

Help

Logout

**Favorite Events**

Mitch Brown

Day Week Month

1001 801

**Cultural Events**

Time	Event
12:00pm	Chamber Music Noon Concerts
8:00pm	Benjamin, Chin and Messiaen
8:00pm	World Percussion Group Concert
8:00pm	Benjamin, Chin and Messiaen
10:15pm	Musical Memories
12:30pm	Patricia Rebaud and William Welborn in Concert
7:30pm	Virtuoso Violin
8:00pm	Kirov Orchestra in Concert
8:00pm	Woodwind Chamber Music

**TV**

6:30pm ABC News

← 507

Add all checked Events to your Calendar [Edit]

807 804

904

903

1001

807 804

In Favorite Events you will see the Event Schedules that you are currently Tracking. To get started or to add Event Schedules:

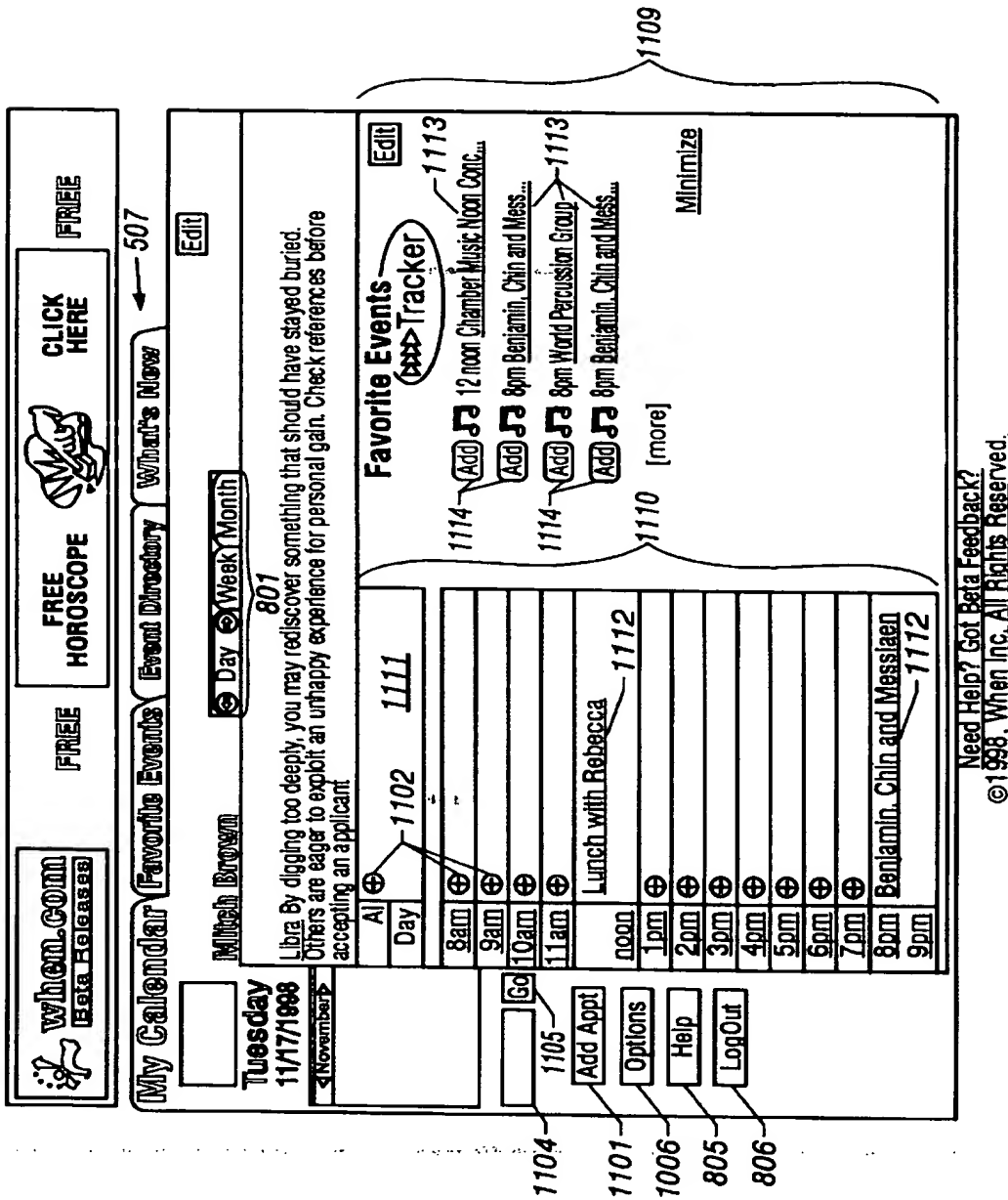
- Visit our Event Directory and browse the categories listed
- Find Event Schedules that interest you and click the Tracker button
- View the Schedules you selected here

Tip: If this page is blank, then either you have not yet selected any Event Schedules to track or no Events exist for this date. If you have not yet selected any Events Schedules to track, use the Event Directory.

Add all checked Events to your Calendar [Edit]

Need Help? Got Beta Feedback?  
©1998, When Inc. All Rights Reserved.

FIG. 10



Need Help? Got Beta Feedback?  
©1998, When Inc. All Rights Reserved.

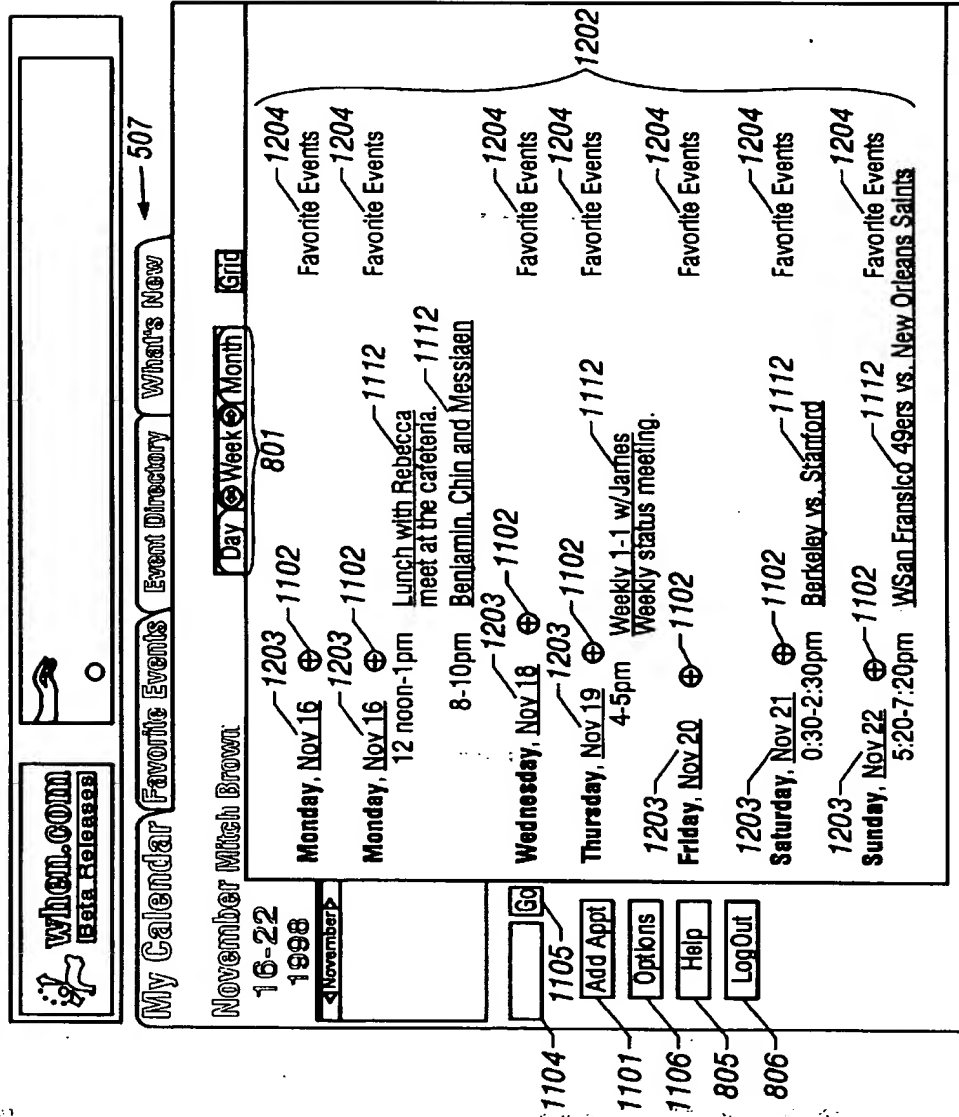


FIG. 12

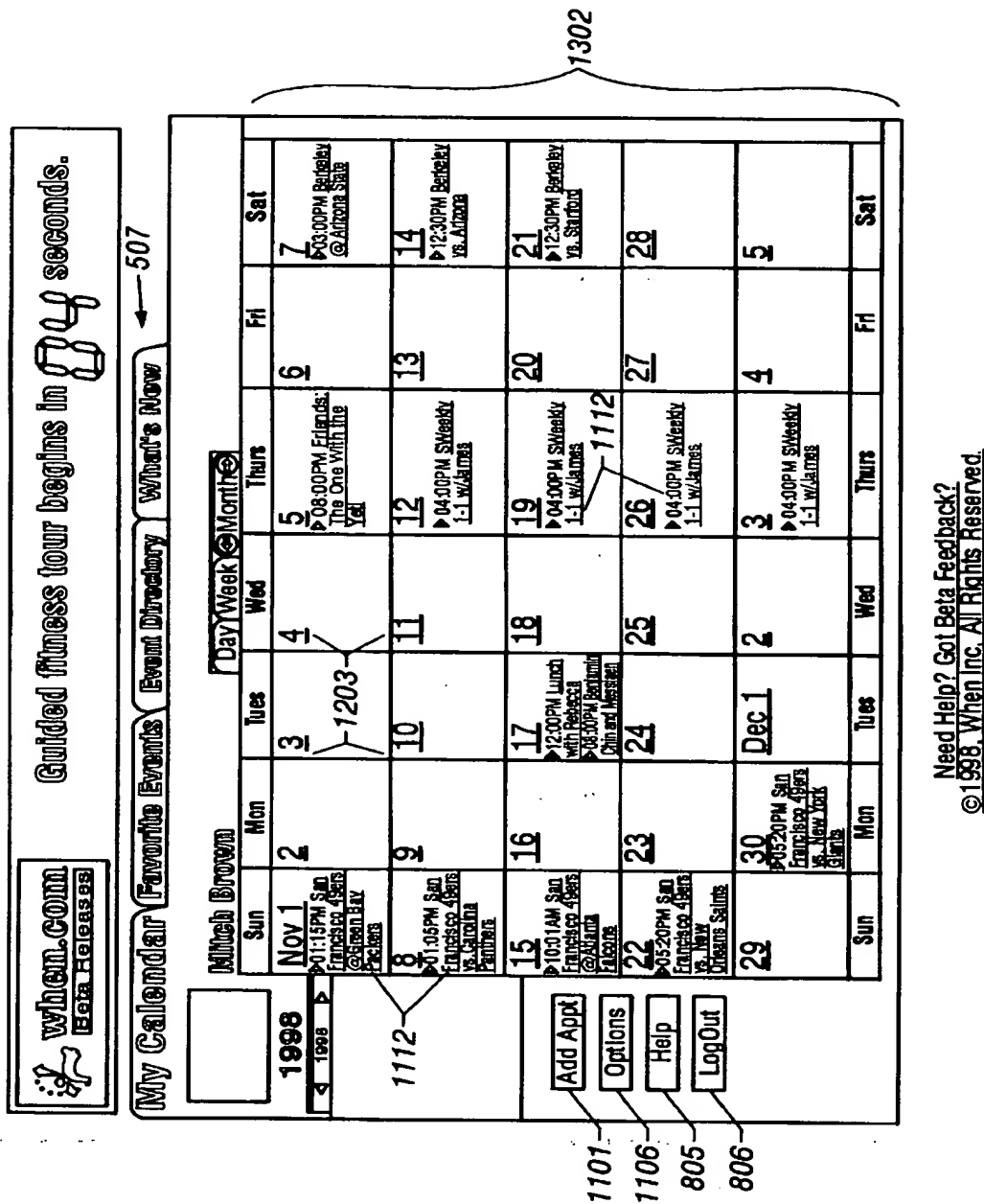




FIG. 13

311 →

  
DATA RELEASES

FREE

HOROSCOPE

CLICK HERE

FREE

**My Calendar**

[Help](#)

**Title** 1402

Lunch with Rebecca 1403

**Date** 1403

11/17/1998 1404

**Time** 1404

12pm ▾ :00 ▾ 1405

**Duration** 1405

☒ :00 ▾ Hours ▾ :00 ▾ Minutes 1409  
☐ All Day 1406 1411

**Notes** 1407

**Options** 1408

☐ Recurs: Daily 1410  
 Start Date: 1  
 End Date:

☐ All Day for ▾ Days in a Row 1413

Save Appointment 1412

Back 1412

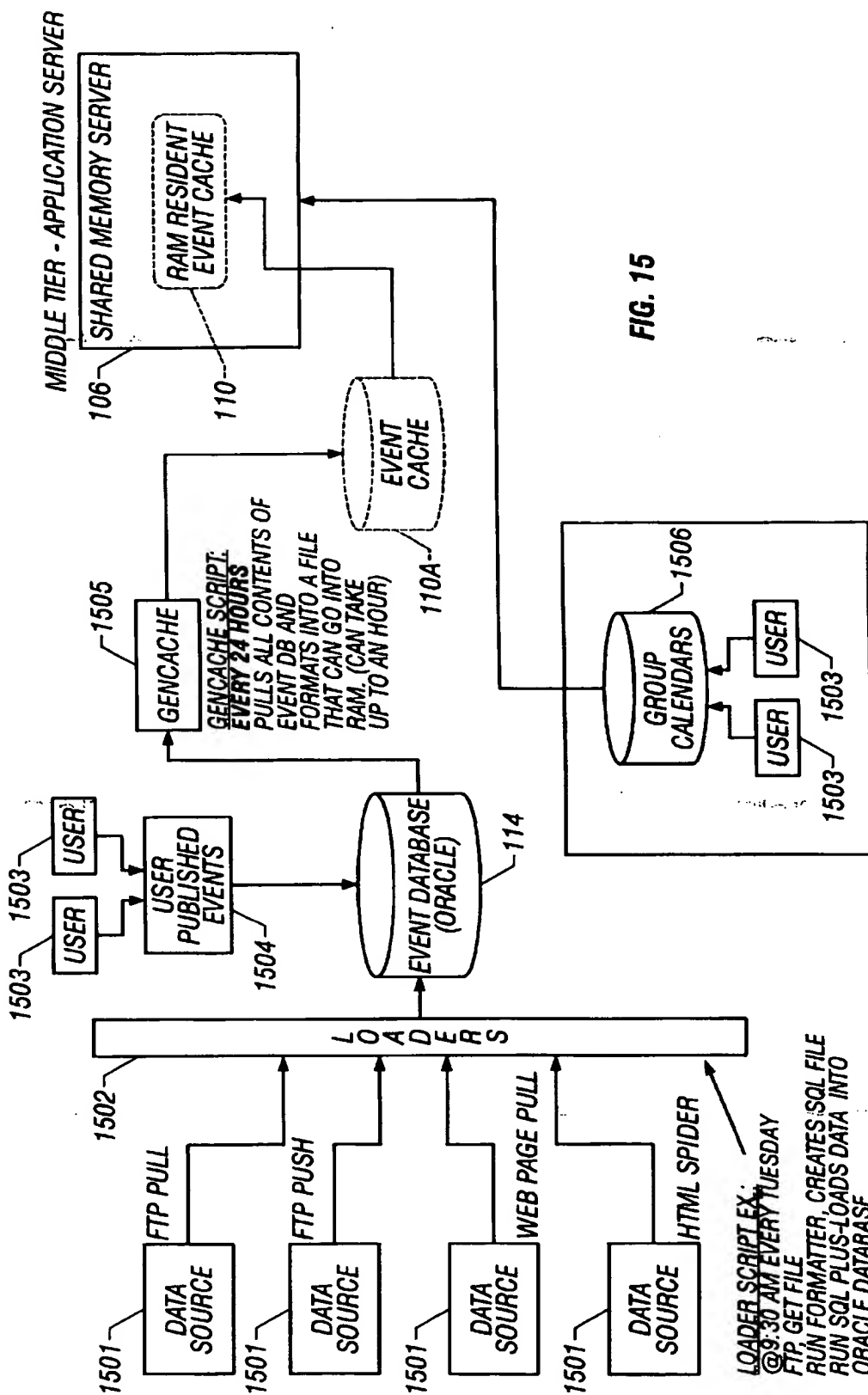
Delete Appointment 1413

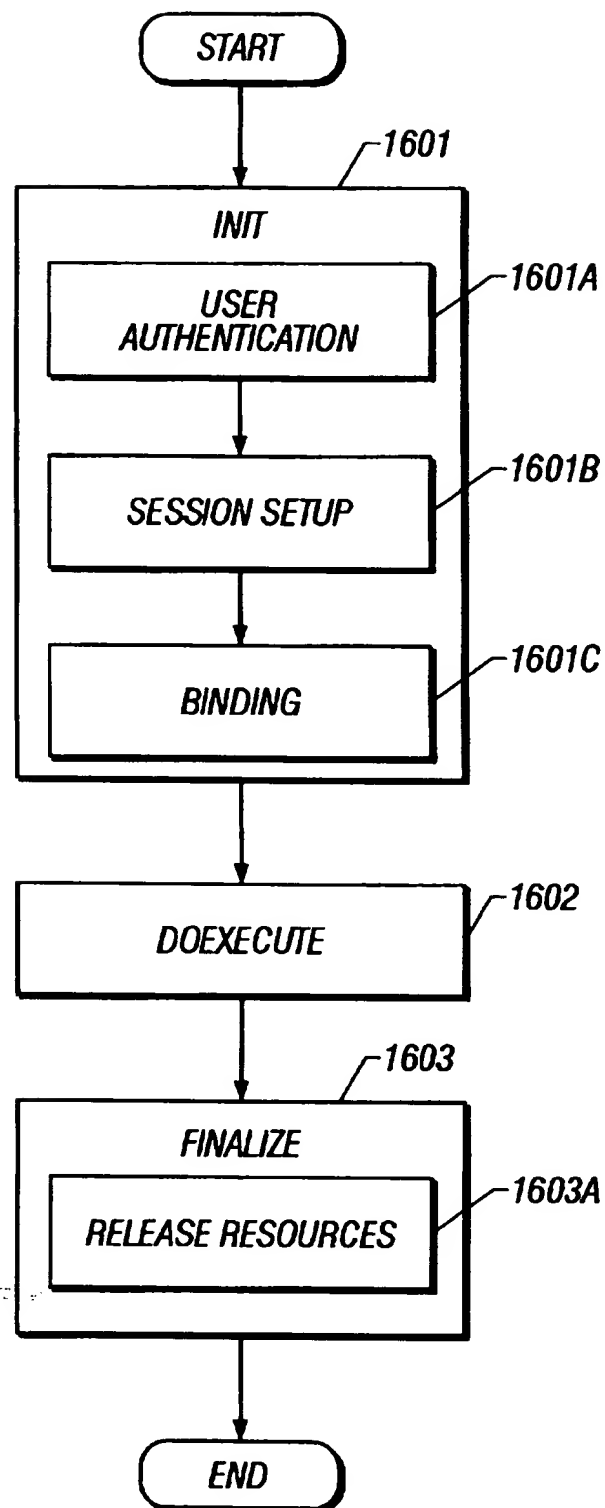
101

FIG. 14

Need Help? Got Beta Feedback?  
 ©1998, When Inc. All Rights Reserved.





**FIG. 16**

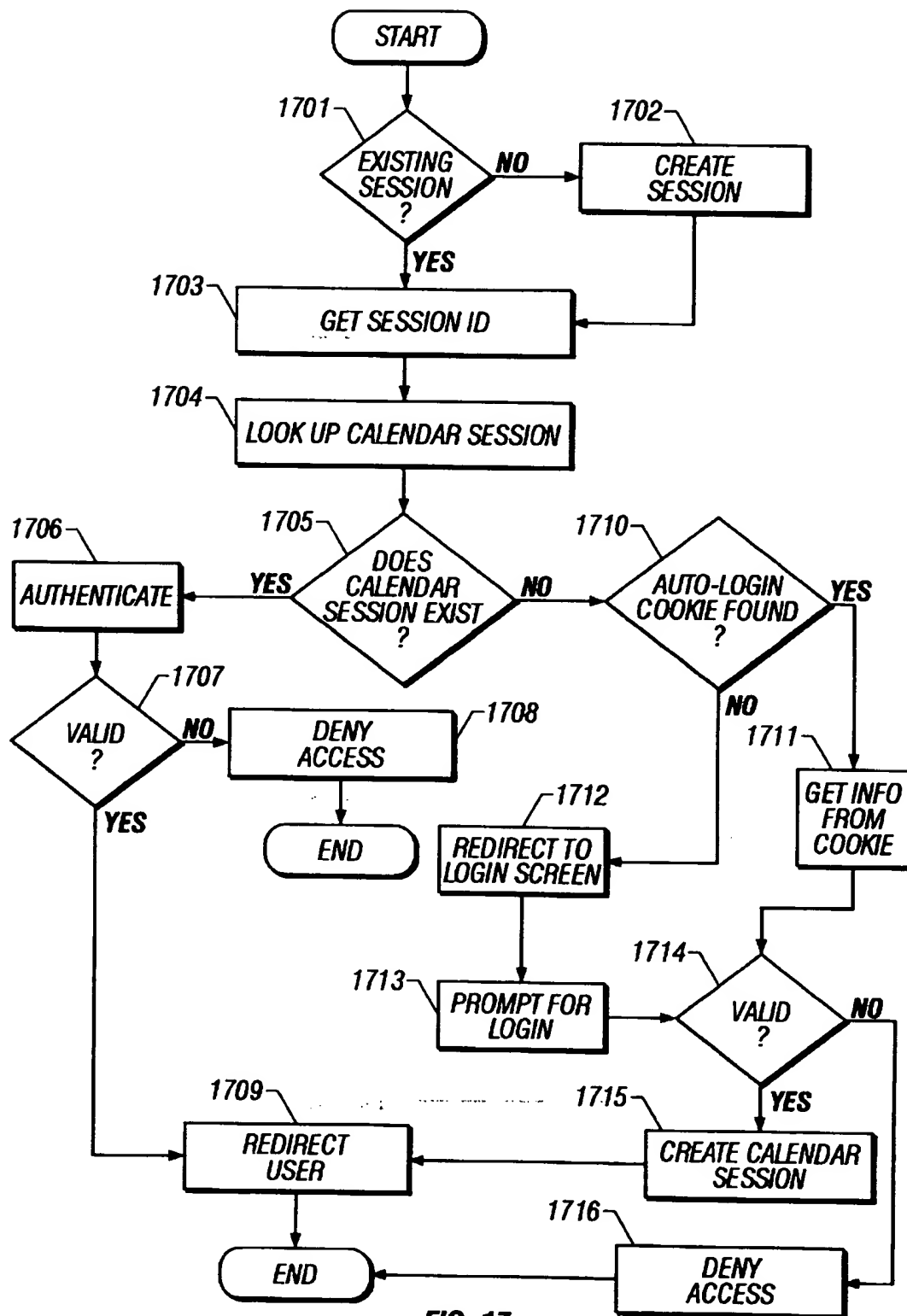
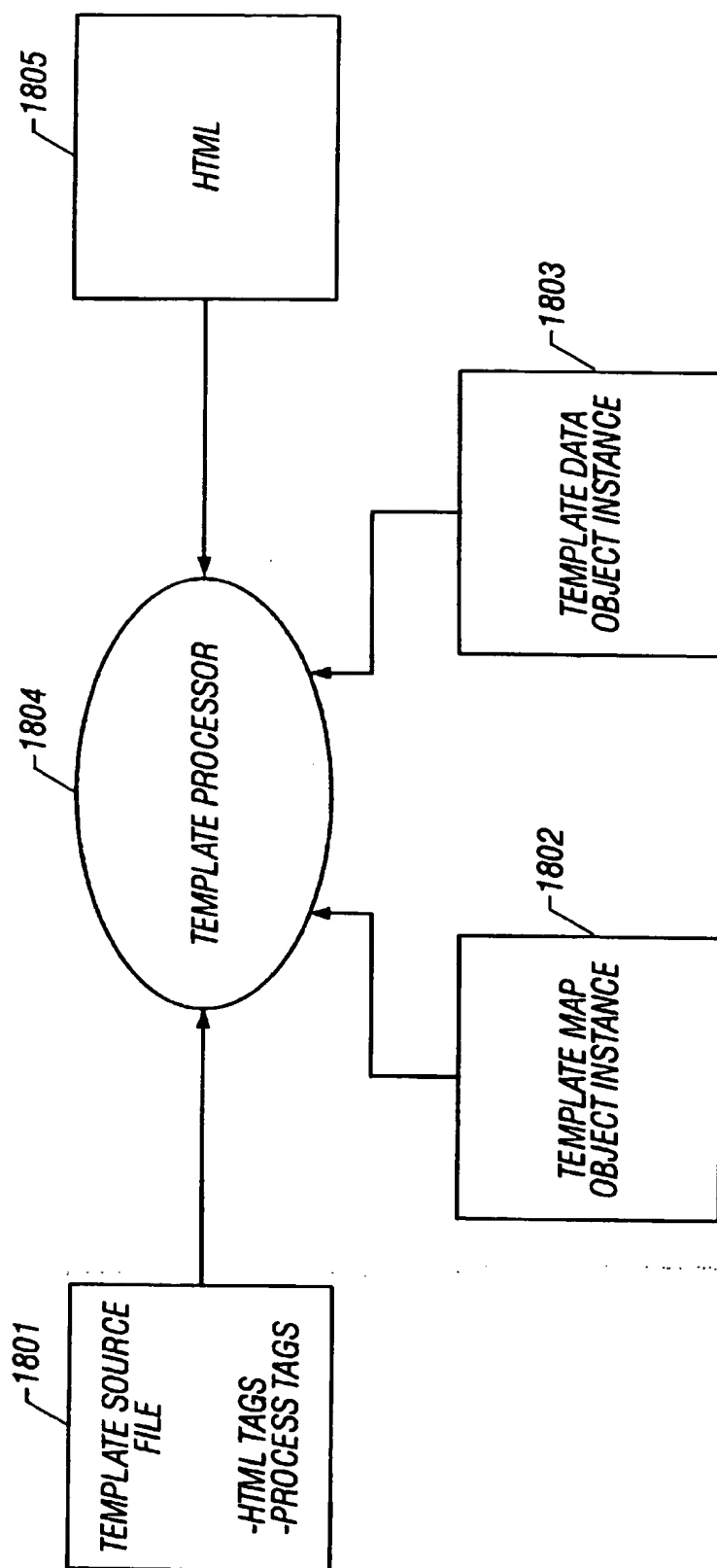
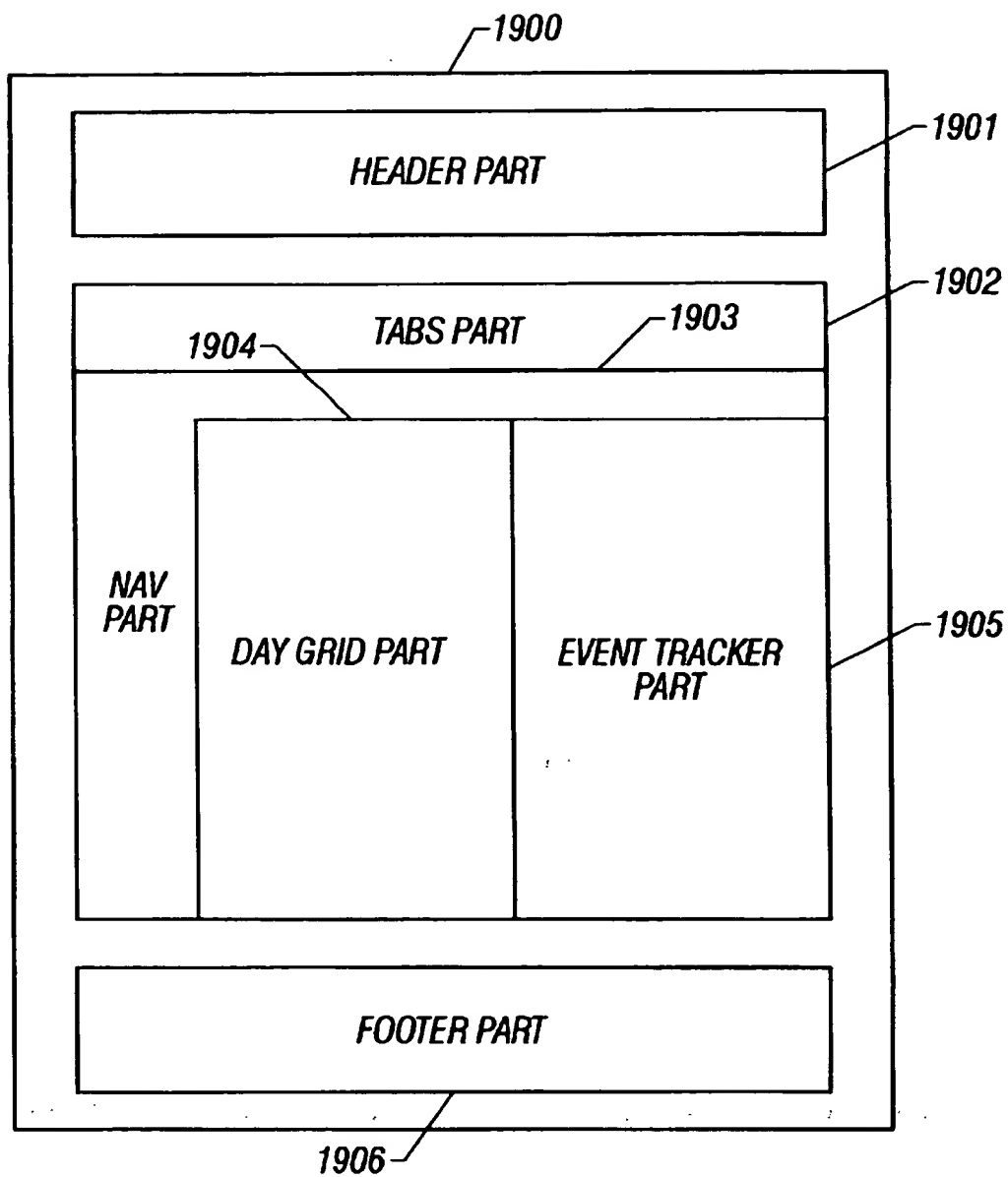


FIG. 17

**FIG. 18**

**FIG. 19**

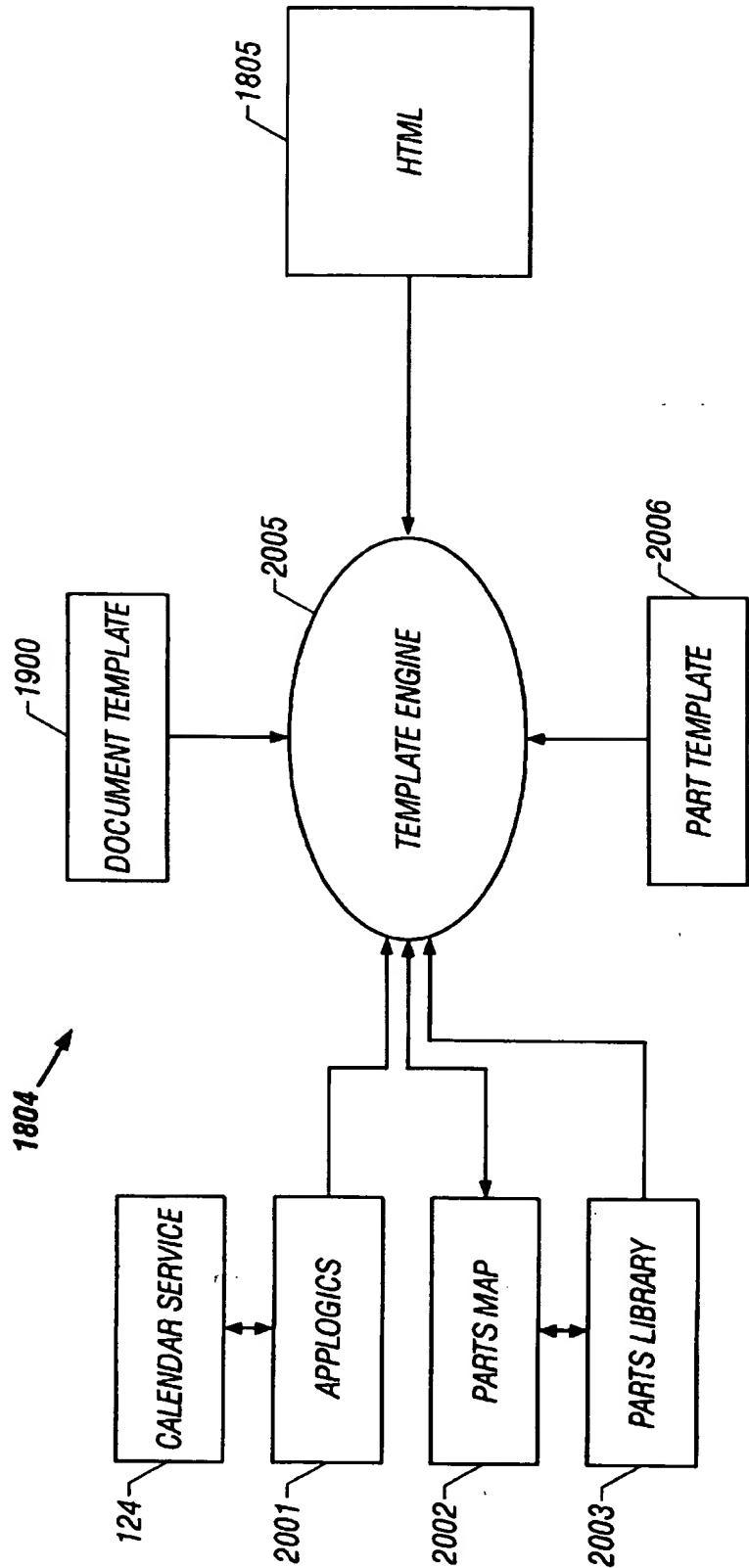
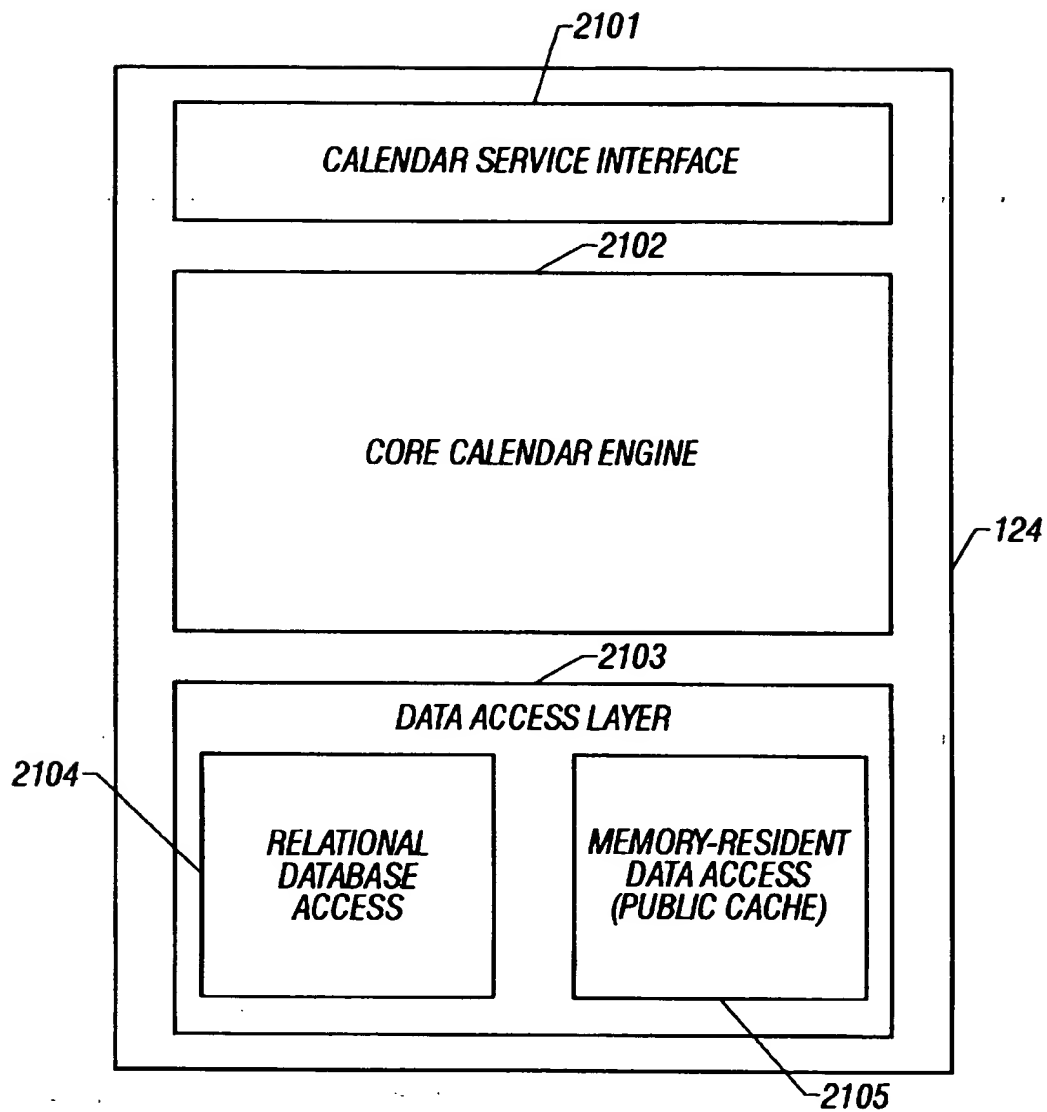


FIG. 20

**FIG. 21**

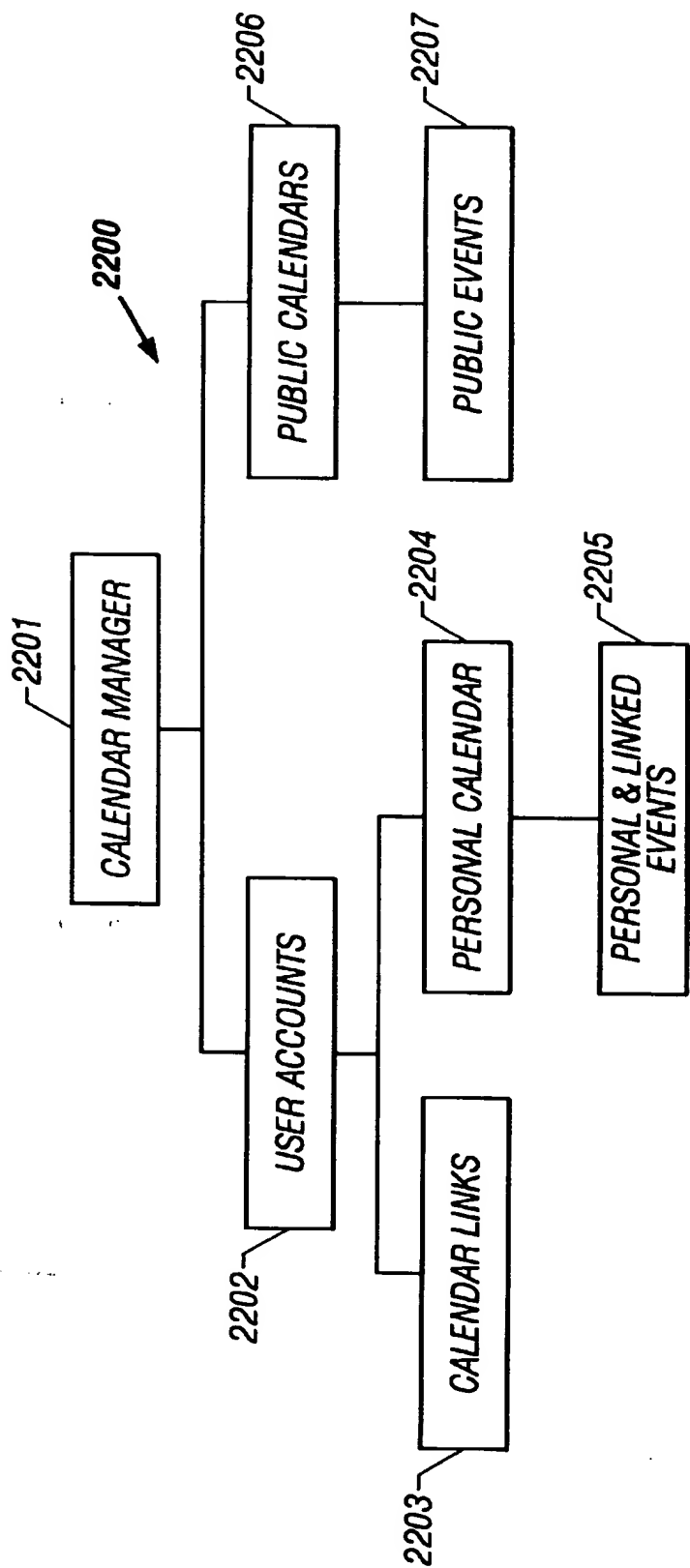


FIG. 22



1

## MULTI-LAYERED ONLINE CALENDARING AND PURCHASING

### BACKGROUND OF THE INVENTION

#### 1. Field of the Invention

The present invention relates generally to software for generating and manipulating computer-implemented planning calendars, and more particularly to a system and method of multi-layered online calendaring and purchasing.

#### 2. Description of Background Art

Conventional software applications for calendaring and scheduling generally take one of two forms: stand-alone or networked. In a stand-alone calendaring application, some sort of user interface is provided which allows a user to specify dates and times for events. Events may then be viewed on a day-by-day, week-by-week, or month-by-month basis, depending on the user's wishes. In many such applications, selective viewing of certain predefined categories of events is provided.

An example of a stand-alone calendaring application is "In Control" by Attain Corporation. Many other calendaring applications are available which offer similar features. In "In Control", three views of the user's calendar information are available: Outline, Calendar, and Day. Outline View allows the user to view events in a list format, and also permits hierarchical arrangement of the event descriptions. A date and time can be associated with each event, and the user can also specify other types of information, such as priority, description, and the like. In Calendar View, a conventional calendar is displayed. Events from the Outline View are shown on the calendar in their appropriate locations according to the date associated with each event. The user can specify the range of dates to be displayed, such as for example one week, two weeks, or one month. In Day View, events for a single day are shown on a display resembling a conventional paper day-planner. Items having a specific time are shown within the daily schedule at the appropriate time. Events not having a specific time are listed in a "To-Do" list next to the scheduled events.

Stand-alone calendaring applications, such as "In Control", are effective for managing one's calendar, but they do not provide an easy mechanism for importing events from an outside source in an automated manner. For example, if a user is planning to attend a baseball game at 7:30 p.m. on Thursday, the entry for the baseball game must be manually entered in the calendaring application. Such applications do not provide an easy way to automatically import such information from, for example, a list of sporting events from an outside source. The following disadvantages present themselves:

Manual entry of events is prone to errors;

If the scheduled event changes, the user may not be aware of the change and may fail to reflect it in the calendar; and

The above-described method does not provide a way to inform the user of events that may be of possible interest.

In addition, stand-alone applications do not provide the capability of sharing one's calendar information with other users.

Some prior art calendaring applications operate in a networked environment and thereby allow sharing of calendar information. Individuals' calendar information can be shared across a network connection (if the owner of the information grants permission for such access). Applications

2

such as Microsoft Outlook, from Microsoft Corporation, provide this type of functionality. However, such applications do not generally provide the ability to import event information from outside sources on a category-by-category basis, and then to select individual events from selected categories for inclusion in a user's personal calendar. Furthermore, such applications do not provide a multi-layered calendaring system wherein events belonging to different categories and selected by a user can be overlaid on one another in a single integrated calendar.

Relatively recently, hosted calendaring applications have been developed which store, in a central location, all calendaring information for a large number of users. Such prior art systems include, for example, appoint.net (www.appoint.net), Yahoo! Calendar (calendar.yahoo.com), and EventCenter from Amplitude Software Corp. (www.amplitude.com). Users access their calendar information across a network, such as the Internet, and security is assured by requiring that each user provide a login and password when accessing the system.

Some of these hosted calendaring systems allow users to add events from outside sources to their personal calendars, if desired. In general, however, such capability is limited in its flexibility. In particular, none of these calendaring systems allow a user to select a category of events, and subsequently add individual events from the category to a personal calendar. Furthermore, none of these systems provide a multi-layered calendaring system wherein events belonging to different categories and selected by a user can be overlaid on one another in a single integrated calendar.

What is needed is a calendaring application that allows a higher level of flexibility in the way events can be imported and viewed.

What is further needed is a calendaring application that permits a user to select categories of events that are of interest, and which provides features allowing a user to add selected events from those categories to his or her personal calendar.

What is further needed is a calendaring application that allows a user to associate layers with certain subsets of events, and to selectively view any desired combination of layers in an integrated manner on a personal calendar page.

What is further needed is a calendaring application that allows a user to share selected calendar information, including selected events, with other users.

What is further needed is a calendaring application that allows a user to purchase products, services, or tickets associated with an event, using on-line communication means.

### SUMMARY OF THE INVENTION

In accordance with the present invention, there is provided a multi-layered online calendaring and purchasing system and method which allows a user to specify categories of events, to view events belonging to the specified categories from outside sources, and to add selected events from the outside sources to a personal calendar. The user can choose which categories of selected events are to be displayed, in any combination he or she desires. The user's personal calendar can also be shared with other users, or selected events and/or categories can be shared, as desired. The user can set up a group calendar, specifying the members in the group, where every group member can access the calendar and make changes to it. Different levels of access can be specified for different members of the group. The user can also import events from other users' calendars. In addition, purchases of products, services, or tickets can be effected using links associated with displayed events.

3

A user logs onto the calendaring system by providing a unique login name and password that identifies the user and allows the system to retrieve that user's personal calendar and associated information. In one embodiment, as described below, the calendaring system is hosted on a server that is connected to the Internet, and the user logs in by interacting with the server via a web page.

Once the user has logged in, he or she can enter any of several different areas of the system, in order to perform different types of activities. An Event Directory allows the user to select categories of events that are of interest. An Event Tracker allows the user to view events associated with selected categories, to obtain more details concerning such events, and to selectively add events to the user's personal calendar. A My Calendar area provides several views of the user's personal calendar, including events that were selected using the Event Tracker, as well as events that have been manually added by the user. In the My Calendar area, the user is able to view and manipulate any of these events. Finally, a What's New area is available for alerting the user to new categories and events that may be of interest; this area may also be used to emphasize particularly important events.

The Event Directory provides listings of event categories, preferably arranged by area of interest. Event categories include, for example, movie opening dates, sporting events, computer tradeshows, and the like. Users can develop their own event categories and share them with other users by publishing them on the Event Directory. The user can click on any event category and view a list of events belonging to that category. Additional information can be obtained for each of the events. The user can choose to select any event categories for inclusion in the Event Tracker. The user can also select a "localized" option which restricts events to those located in the user's city, state or other region.

The Event Tracker displays a list of events belonging to the selected categories. The user can select from several different views of the displayed events and can also choose to view one category at a time, or all categories at once. Events can be sorted by date or by category, as desired. The user can click on a button to add a particular event to his or her personal calendar. In addition, the user can click on a button for online ordering and purchasing of products, services, or tickets as appropriate for the particular event.

The My Calendar area provides an extremely flexible and configurable personal calendar. The user can choose from daily, weekly, or monthly views, and can select particular categories of events to be displayed, or can choose to see all events. Thus, the system provides a multi-layered calendar, where each layer corresponds to an event category. Viewing multiple categories simultaneously provides an integrated personal calendar showing all events in one place. The user can add appointments and other events manually in the My Calendar area, and such events are displayed alongside events that were selected in the Event Tracker. The user can also specify that he or she would like to be notified when an event is about to occur, either by e-mail or by some other communications means. Finally, the user can specify whether he or she would like to share the personal calendar with other users.

### BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of the overall architecture of an embodiment of the present invention.

FIG. 1A is a block diagram of an application server according to the present invention.

4

FIG. 1B is a block diagram of the services implemented in one embodiment of an application server according to the present invention.

FIG. 2 is a block diagram of an embodiment of a client computer for practicing the present invention.

FIG. 3 is a functional block diagram of a block diagram of the functional components of the user interface of the present invention.

FIG. 4 is a screen shot of a login page.

FIG. 5 is a screen shot of a What's New page.

FIG. 6 is a screen shot of an Event Directory page.

FIGS. 7A and 7B are screen shots of an example of an event schedule in month view.

FIG. 8 is a screen shot of a Month View of a Favorite Events screen.

FIG. 9 is a screen shot of a Week View of a Favorite Events screen.

FIG. 10 is a screen shot of a Day View of a Favorite Events screen.

FIG. 11 is a screen shot of a Day View of a My Calendar screen.

FIG. 12 is a screen shot of a Week View of a My Calendar screen.

FIG. 13 is a screen shot of a Month View of a My Calendar screen.

FIG. 14 is a screen shot showing a My Calendar detail screen.

FIG. 15 is a block diagram showing the collection of events data according to the present invention.

FIG. 16 is a flowchart showing basic operation of a default Execute method.

FIG. 17 is a flowchart showing a user authentication process.

FIG. 18 is a block diagram of the operation of a template processor to generate an HTML file.

FIG. 19 is a diagram showing a document template having several parts.

FIG. 20 is a block diagram of the detailed operation of a template processor to generate an HTML file.

FIG. 21 is a block diagram of a calendar service according to one embodiment of the present invention.

FIG. 22 is a block diagram showing an example of an object ownership hierarchy.

### DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

#### System Architecture

Referring now to FIG. 1, there is shown a block diagram of the system architecture of an embodiment of the present invention. System 100 is implemented in a networked computing environment. Individual elements communicate with one another using standard protocols such as, for example, Transfer Control Protocol/Internet Protocol (TCP/IP) and Hypertext Transfer Protocol (HTTP), over a data communication line such as a T1 or T3 line. Other implementations are also possible, and system 100 may even be implemented on a non-networked computer, if desired.

In one embodiment, the present invention operates in a networked environment such as the Internet or an intranet, and pages are provided for user access and interaction via a browser over the World Wide Web 116, as is known in the art.

Director 101 connects an individual client computer (not shown) to system 100. Director 101 handles the low-level

5

interaction with an individual client computer, and accepts input and output for transmission to and from web servers layer 102. In one embodiment, where several load balancers 105 are provided within layer 102, director 101 selects the least loaded load balancer 105 for connection, so that the load is relatively balanced among the components of web servers layer 102. Director 101 is also able to invoke a thread to determine if the user is a new or returning user, based on either 1) the particular Uniform Resource Locator (URL) string used to connect to system 100; or 2) a "cookie" file that has previously been stored on the user's machine. If the user is a returning user, director 101 can call a What's New page which retrieves personal calendar information from database servers layer 104 and presents relevant information to the user as appropriate.

In one embodiment, director 101 is implemented using Big IP, from F5 Labs, Inc., for connecting web servers layer 102 to the World Wide Web 116 in a multiplexed manner for optimal load balancing.

Web servers layer 102 contains one or more load balancers 105 for determining which application server 106 is best able to handle a particular connection for a particular user. In one embodiment, load balancers 105 are implemented as Sun Microsystems Ultra 5 servers running a load balancing client. New users are sent to the least-loaded application server 106. Returning users are sent to the application server 106 containing the process that serviced the user during the most recent connection with that user. This technique advantageously facilitates access to a previously stored user cache, which improves performance as will be described below in connection with FIG. 1A. A long-term memory (LTM) 107 provides a mechanism for storing information as to which process serviced the user in the past, so that this information can be retrieved by load balancers 105 upon the user's return.

Application servers layer 103 contains one or more application servers 106. Application servers layer 103 provides an intermediate layer between database layer 104 and web server layer 102. Application servers 106 run the software code for interacting with database layer 104 and for retrieving, modifying, and storing data in databases 112 and 114. In one embodiment, each application server 106 is a Sun Microsystems Enterprise 450 server.

Database servers layer 104 contains databases such as personal calendar information database 112, events database 114, and the like, for maintaining scheduling and other information for users of the system, and also maintaining information describing scheduled events and announcements. In one implementation, individual databases within layer 104 are stored on separate database servers such as Sun Microsystems Enterprise 4500 servers. Databases may be stored in parallel redundant fashion for backup purposes. Personal Calendar Information database 112 contains various types of data, including personal event data 111 for various users. Events database 114 can also connect with other data sources (not shown) as desired, so that information describing events can be imported and made available to users of system 100.

In an alternative embodiment, each database in database servers layer 104 can be split into a plurality of smaller databases, each for a subset of users.

Referring now to FIG. 1A, there is shown a block diagram of an application server 106 according to one embodiment of the present invention. Application server 106 serves personal calendars, event directory contents, user profiles, and other types of pages to users in response to Hypertext Transfer Protocol (HTTP) requests relayed by load balanc-

6

ers 105 in web servers layer 102. In one embodiment, server 106 is implemented as a set of shared libraries that are dynamically linked and loaded at runtime.

Application server 106 is capable of running a number of processes 108 simultaneously. Upon initial connection by a particular user, the user is assigned to a selected process 108, based on load balancing parameters.

Each process 108 contains a number of simultaneously-executing application threads 115. Application server processes 108 interact with user cache 109, which provides temporary storage of personal calendar information for particular users. Such information may include, for example, the user's selected settings and options, favorite events, and other information. Cache 109 obtains information from personal calendar information database 112 as needed, and writes information to database 112 when appropriate. Cache 109 provides improved performance by obviating the need for a direct connection between process 108 and database 112. In one embodiment, cache 109 for a particular user is released from memory when either some predetermined time period expires, or when memory is needed and that user's data is the oldest cache 109 that has not yet expired.

Process 108 also interacts with event cache 110, which provides temporary storage of event data taken from events database 114.

In one embodiment, process 108 remembers where a particular user's cache 109 is located, so that if the user returns to the site (i.e. re-establishes a connection) and his or her cache 109 is still available, process 108 is able to utilize the cache 109.

Referring now to FIG. 1B, there is shown a block diagram of the services implemented in one embodiment of an application server 106 according to the present invention. Application implementation 121 implements each page of the user interface using templates and a component-driven user interface implementation. Application implementation 121 also parses HTTP parameters and generates HTML output for pages. Utilities 122 include implementations for parsing and formatting functions, high-level calendar operations, and e-mail notification services. Session management 123 authenticates, tracks, and manages user sessions for calendaring operations, and stores a cache of user and calendar data as needed to maintain sessions. Calendar service 124 implements the core object model for accessing user profiles, calendars, events, calendar links, and collections of objects. Service 124 also implements event lookup, object caching, and persistence. Finally, service 124 serves both personal calendars and event directory data using a common object model. Other services 125 provide access to non-calendar-oriented data services such as weather, horoscope, and application configuration settings. The operation of the relevant components of application server 106 to implement the present invention will be described in more detail below.

Client Computer

Referring now to FIG. 2, there is shown a block diagram of a client computer 200 that can be connected to system 100 for practicing the present invention. In one embodiment, client computer 200 is implemented on a personal computer running the Microsoft® Windows™ 95 operating system on an Intel® Pentium® processor. The user interacts with the present invention by establishing a network connection with director 101 over the World Wide Web 116 using a TCP/IP connection and a browser application running on the aforementioned client computer 200. Thus, as the user operates the present invention, he or she is presented with interactive web pages that provide information and accept input, as is

known in the art. One skilled in the art will recognize that other embodiments, including other types of software applications, processors, and operating systems, are also possible.

In the block diagram of FIG. 2, client computer 200 is shown having a central processing unit (CPU) 201, display device 202, input device such as a pointing device 203, random access memory (RAM) 204, and storage device 205. Also provided is a connection 207 to a network such as the World Wide Web 116, in order to establish contact with system 100 of the present invention. The following detailed description of the invention will make reference to exemplary implementations of such components, though other embodiments may also be used. For example, CPU 201 is a microprocessor such as an Intel Pentium processor; display device 202 is a conventional monitor or screen such as a cathode-ray tube (CRT); pointing device 203 is a mouse or trackball; though other input devices such as keyboards can also be used; RAM 204 is some quantity of conventional memory as is commonly supplied with personal computers; storage device 205 is a hard disk or similar device for long-term storage of programs and data; and Internet connection 207 is implemented using known protocols such as TCP/IP across a modem, T1 or T3 line, or other connection medium.

In a preferred embodiment, the user interacts with system 100 using a browser application. Such browsers are well known in the art, including for example Netscape Navigator and Microsoft Internet Explorer. One skilled in the art will recognize that other embodiments of the invention, that may operate without use of a browser, are also possible.

For purposes of this description, the terms "page" and "screen" are used interchangeably to refer to a user interface element presented to the user via the browser.

#### User Interface and Application Operation

Referring now to FIG. 3, there is shown a block diagram of the functional components of the user interface of the present invention. Each of the functional components shown in FIG. 3 will be described in more detail below, with reference to specific screen shots and other user interface elements. As will be seen below, each element in FIG. 3 may be implemented as a web page in an Internet-based application. The user can navigate among the various pages shown by clicking on links in hypertext documents, as is known in the art of browsing web pages. As will be shown in other Figures, one embodiment of the present invention uses a metaphor showing "tabs" for navigation among the pages.

FIG. 3 also illustrates the overall task flow 300 of one embodiment of the present invention, with connections between elements representing hypertext links from one page to another.

Home/login page 301 welcomes the user to the application or website, upon initial connection. The user may elect to login at this point by providing input specifying a login identifier and password. This allows system 100 to retrieve user-specific information, by reference to a record stored in database 104 of the system.

If the user has not used the system before, he or she is prompted to sign up in 302, by selecting a login identifier and password for future reference. A new record is created and stored for the user. The user is also given the option of signing up in a group using the group sign-up page 304, which allows the user to share his or her calendar with other members of selected groups. Page 303 contains a description of groups and their operation.

Once the new user has signed up using 302 and/or 304, a welcome page 305 is presented, confirming that the user's record has been stored in the system.

Once the user has been identified via user login 301, or signed up and welcomed (302, 304, and 305), a What's New page 306 is presented. As will be described in more detail below, What's New page 306 highlights important events and announcements that may be of interest to the user. New event calendars may also be presented in this screen. In addition, in one embodiment of the present invention wherein shared group calendars are employed, the What's New page 306 can inform users of changes to shared calendars. Finally, the system can display reminders of upcoming events which the user has previously selected.

Event Directory pages 317-319 provide a directory of event categories including Event Directory Contents 317 containing an overall list of event categories, Event Category Home Pages 318, containing descriptions of each of the event categories, and Event Category Subdivisions 319, containing descriptions of subdivisions within particular event categories.

From any of pages 317-319, the user can view schedules of events belonging to a selected category or subdivision. Such event schedules can be viewed in several different formats, such as for example a Day View 320, a Week View in grid format 321, a Week View in list format 322, a Month View in grid format 323, and a Month View in list format 324. Each of these views provide a display of a number of events belonging to a particular category or subdivision. The user can click on a particular event in any of views 320-324 to see an Event Details page 325 showing details for the selected event.

The user can select individual event categories and/or subdivisions for display in Favorite Events pages 313-315. Selecting an event category in this manner is referred to as "subscribing" to the event category. Favorite Events pages 313-315 display selected events in either a Day View 313, a Week View 314, or a Month View 315. Pages 313-315 allow a user to select individual events from the selected categories, to be added to the personal calendar. The user can also access an Edit Favorites page 316 which allows him or her to add or remove categories and/or subdivisions from display in favorite Events pages 313-315.

My Calendar pages 307-310 provide access to the user's personal calendar. My Calendar pages 307-310 show an integrated, multi-layer overview of events from selected categories, as well as manually-entered events. Several displays are available, including a Day View 307, a Week View in list format 308, a Week View in grid format 309, and a Month View 310. Details on any appointment or event are available from the Details page 311. In one embodiment, system uses Event Details page 325 to show details of a selected event that was previously selected from the Event Directory, and Appointment Details page 311 to show details on manually-entered events and appointments. An Options page 312 is also provided, for configuring and selecting among various system options and preferences. Events and appointments can be added, modified, or deleted as desired.

Help pages 326 are provided to assist the user in using the various components of the system. The user can retrieve a selected help page 326 by clicking on a "Help" link on any of the pages in the system.

Referring now to FIG. 4, there is shown a screen shot of a Login page 301 according to one embodiment of the present invention. As with all pages discussed herein, Login page 301 is presented as an HTML page that may be displayed to the user on a conventional browser screen, though other embodiments may use different techniques for presenting user interface elements to the user. Login Name field 402 and password field 403 are provided for the user to

enter relevant information allowing the system to identify him or her. In response to the user entering the information, system 100 retrieves centrally stored user-specific information 111 from database 112, including user preferences and personalized calendar information. As described above, such information is retrieved from user cache 109 if available.

Login page 301 also provides links to other pages, for initial sign-up 404, on-line help 407, event categories 406, and miscellaneous information about the service 405.

Referring now to FIG. 5, there is shown a screen shot of a What's New page 306 according to one embodiment of the present invention. A personalized welcome greeting 501 is displayed, as well as any important announcements 502. The information displayed in What's New page is taken from the user's individual records in database 112.

In an alternative embodiment, the What's New page provides information describing changes to shared calendars, new features, and other relevant information. Reminders of upcoming events which were previously selected by the user can be provided as well.

Navigation bar 507 provides links to other pages in system 100, including My Calendar pages 307-310, Favorite Events pages 313-315, and Event Directory pages 317-319. Buttons 503 and 504 provide access to Event Directory pages 317-319 and My Calendar pages 307-310, respectively. Links 505 provide access to a Group Calendar feature, as described in more detail.

In one embodiment, a list of new event categories is provided (not shown). This list may include any categories that have been added to system 100 since the user last logged on, or it may be a list of new categories that are related to other categories to which the user has previously subscribed. Thus, system 100 is able to provide event category suggestions that are likely to be of interest to the user, based on his or her previous behavior. The user can select one or more event categories from the list. If desired, the user can obtain additional information on event categories.

Page 306 shown in FIG. 5 is one example of a What's New page according to the present invention. Other configurations of the What's New page are also possible, without departing from the spirit or essential characteristics of the present invention. In particular, additional information can be provided on this page, such as announcements regarding changes to event categories to which the user has previously described. Also, the What's New page may be configurable, so that the user can select what kind of information is displayed there.

Referring now to FIG. 6, there is shown a screen shot of an Event Directory Screen 317. Hyperlinks 601 to event categories are displayed, allowing the user to obtain more information, and if desired, "subscribe" to that event category. Each hyperlink 601 links to an Event Category Home Page 318 providing detailed descriptions of event categories. Each event category is a group of related events to which the user can subscribe, if desired. Event categories can include, for example, sports team schedules, movie release dates, schedules of conferences on a particular topic, meeting schedules related to a particular project or company, and the like.

Event categories and related information describing individual events are provided to system 100 and stored in events database 114. Such information may be provided by external sources, such as online services listing scheduled events, as described below in connection with FIG. 15. Alternatively, such information may be manually entered in events database 114 by a system operator. In one embodiment of the present invention, individual users may provide

user-defined categories in the form of group calendars including scheduled events the user wishes to share with other users. For example, a project leader may create a personal calendar including meetings and deadlines related to the project, and then share that personal calendar as a group calendar. Other users involved with the project may then subscribe to the group calendar as an event category. If desired, security measures may be provided restricting access to a group calendar, so that only selected users can retrieve the information.

In one embodiment, a search field (not shown) is provided to allow interactive searching of the event directory.

Referring now to FIG. 15, there is shown a block diagram illustrating the collection of events data according to one embodiment of the present invention. The process performed by the apparatus of FIG. 15 receives event feeds from content partners, processes the event data, and stores them in event database 114. Information from database 114 is cached into shared memory for access by application server processes as needed. The elements of FIG. 15 are implemented as a collection of programs and scripts for automated operation and import of event data.

Loaders 1502 are software scripts resident in database servers layer 104 for extracting events data from external data sources 1501 such as web pages, publicly-accessible files and databases, and the like. In one embodiment, loaders 1502 are configured to extract such data on a regular basis from a predefined set of sources 1501. For example, a loader 1502 might be configured to run a File Transfer Protocol (FTP) operation to obtain data from a source 1501 at 9:30 a.m. every Tuesday morning, then to run the obtained data through a formatter to create a Structured Query Language (SQL) file, and finally to run a script which loads the data from the SQL file into events database 114. Loaders 1502 may obtain information from data sources 1501 by any conventional means, such as for example: FTP pull (FTP operation initiated by loader 1502); FTP push (FTP operation initiated by data source 1501); web page push (Hypertext Transfer Protocol (HTTP) operation initiated by data source 1501); or Hypertext Markup Language (HTML) spider (automated traversal through a series of HTML pages on the World Wide Web).

Once loaders 1502 have collected information from data sources 1501, the data is formatted and written to events database 114. Users 1503 may also provide user-published events 1504 manually, which are also added to events database 114. In addition, a group calendars database 1506 may be provided for storing events information related to a particular predefined group of users. Users 1503 can contribute event information that they would like to share with other users 1503 in a particular group. Such event information is written to group calendar database 1506 and then transferred to events database 114.

In one embodiment, application servers 106 read and operate on event data using an event cache 110 (see FIG. 1A) in order to provide improved performance. In order to facilitate such operation, a cache generation (GenCache) operation 1505 is provided. GenCache operation reads events database 114 on a regular basis (e.g. daily) and generates an Event Cache file 110A containing selected event data in a format that can be transferred to RAM. In general, it is preferable if Event Cache file 110A contains data that is likely to be accessed, so that such access can be achieved without resorting to additional reads from events database 114. Once Event Cache file 110A has been generated, it is written to event cache 110 in application server 106. Event cache 110 is RAM-resident in server 106

11

and is therefore capable of being accessed more efficiently than can events database 114.

Once event cache 110 is in place, individual processes 108 can access events data from cache 110 as needed. If a particular event datum is not present in cache 110, application server 106 initiates a read from events database 114 to obtain the needed information.

Referring now to FIGS. 7A and 7B, there are shown screen shots of the Event Schedule Month View in list format 324 and in grid format 323. Screen shots 324 and 323 may be displayed, for example, when a user selects a particular event category by clicking on one of the links 601 in the Event Directory screen 317. In screen 324, the user may click on grid button 703 to be transferred to screen 323. In screen 323, the user may click on list button 704 to be transferred to screen 324. Buttons 705 allow the user to view other time periods besides the one being displayed. In one embodiment, the information displayed in screens 324 and 323 comes from events database 114, which in turn may be collected from external sources, as described above in connection with FIG. 15.

Events 706 belonging to the selected event category are shown in list form in screen 324, or in grid form in screen 323. The user can obtain additional detail regarding any listed event 706 by clicking on the link associated with the event 706. System 100 then displays an event detail page (not shown) containing the additional detail.

The user can subscribe to the displayed event category by clicking on Tracker button 702. Events from the event category will then appear on the Favorite Events screens 313-315, as described below. In addition, in one embodiment the user can add individual events to his or her personal calendar without subscribing to the event category, by clicking on a button within the event detail page (not shown), or on a button (not shown) on screen 324 or 323.

Referring now to FIG. 8, there is shown a screen shot of a Month View 315 of a Favorite Events screen according to one embodiment of the present invention. Favorite Events screen allows the user to view events from selected categories to which he or she has subscribed. The user can select whether to view all events for a time period from subscribed categories, or to view events only from selected subscribed categories. Thus, the subscribed categories can be thought of as "layers" which can be placed on top of one another and viewed together, separately, or in any combination. This layering concept extends to views of the user's personal calendar, as will be seen below. Buttons 802 specify individual event categories for display, which the user can select in any combination desired. In one embodiment, buttons 802 are provided for each event category to which the user has previously subscribed. Screen 315 shows all events belonging to the selected event categories in the month being displayed. Update button 803 takes the user to the Event Directory screens 317-319 (see above), where he or she can subscribe or unsubscribe to event categories.

In one embodiment, the Favorite Events screen also contains event categories representing shared group calendars of other users. Thus, users can create personal or work-related group calendars which can be shared and subscribed to as are public calendars. Such group calendars are listed as additional buttons 802, allowing the user to select or de-select individual group calendars are view associated events as desired, "layered" on top of other displayed events in screen 315.

In one embodiment of the Favorite Events screen, several different views of events are available. For example, the user may select from a Day View 313, a Week View 314, or a

12

Month View 315. Navigation bar 801 provides links to the available views. Navigation bar 507 provides links to other screens, as described elsewhere in this disclosure. Additional buttons are provided, including an Edit button 804 which allows the user to change personal preferences, a Help button 805 for providing access to Help pages 326, and a Log Out button 806 which exits the system.

A checkbox 904 is provided for each event 903. The user can add an event 903 to his or her personal calendar by clicking on the associated check-box 904 and clicking on button 807. Other mechanisms may be provided for adding individual events, or groups of events to the user's personal calendar. The particular user interface configuration to be used may depend on the characteristics of the network on which system 100 is implemented.

Referring now to FIG. 9, there is shown a screen shot of a Week View 314 of a Favorite Events screen according to one embodiment of the present invention. As with screen shot 315, the user can select individual events belonging to subscribed categories, for inclusion in his or her personal calendar. The week-long view of FIG. 9 shows one week's worth of events in a columnar format.

Navigation bar 507 provides access to other screens of system 100. Buttons 802-806 provide the same functionality as described above in connection with FIG. 8; namely, these buttons allow the user to specify which event categories are to be displayed (or overlaid on one another) in the Favorite Events screen, and also provide Edit, Help, and Log Out functions. Navigation bar 801 allows the user to navigate among Day View 313, Week View 314, and Month View 315 of the Favorite Events screens.

In the Week View shown, each day of the week is displayed as a column 902. The user can click on the column header to see a Day View 313 for the selected day. Within each column 902, events 903 are listed in a grid format by time of day, with a separate space provided for all-day events. In an alternative embodiment, events 903 can be listed using other formats, such as by category, or alphabetically, for example. As in screen 315, the user can add events to his or her personal calendar by clicking on check boxes 904 and button 807.

Referring now to FIG. 10, there is shown a Day View 313 of a Favorite Events screen. Operation of Day View 313 is similar to that of Month View 315 and Week View 313, described above. Day view 313 shows events associated with a particular selected day.

In Day View 313, events are divided by category, denoted by category headings 1001. Events 903 are displayed for each event category heading 1001. Buttons similar to buttons 802 in Month View 315 and Week View 313 may be provided, allowing selection and de-selection of particular event categories for display. In one embodiment, an event category is only listed if it contains events within the day associated with the current display. In other words, in this embodiment, if there are no Cultural Events occurring on the day being displayed, the event category heading 1001 for Cultural Events will be omitted from the display.

In one embodiment, each event category heading 1001 button (not shown) providing a link to the Event Schedule for the category. This link takes the user to a screen, similar to that shown in FIGS. 7A and 7B, listing events belonging to the specified category. Underneath the heading 1001 is a list of individual events 903 belonging to the category. For each event 903, a date and description is provided, as appropriate. If applicable, a time of day may also be shown. If appropriate and available, a link may be provided to a more detailed description of the event. Finally, button 904 is



also provided for adding the event to the user's personal calendar. Thus, as described above, the user can select individual events belonging to subscribed categories, for incorporation into his or her personal calendar. This provides improved flexibility, in that the user need not add an entire calendar and thereby possibly add events which are not of interest, but can tailor the personal calendar to his or her individual needs. In some embodiments, addition of all events in a category can also be provided without requiring individual selection of each event, by clicking on a button (not shown) associated with the category as a whole.

In one embodiment of the present invention, each subscribed category is color-coded, and individual events belonging to the category are tagged by a color flag matching the color of the category. Thus, in the example shown, the movie releases category may be associated with the color blue, and all movie release events would then include a small blue tag. One skilled in the art will recognize that other visual characteristics could be used to associate events with categories, such as a distinctive icon, font, or other technique.

Referring now to FIG. 11, there is shown a screen shot of a Day View of a My Calendar screen 307 according to one embodiment of the present invention. On this screen, the user is able to view and manipulate his or her personal calendar, containing events that have been previously selected from subscribed categories, as well as events that have been manually added by the user.

Navigation bar 507 provides access to other screens of system 100, as described above. Navigation bar 801 allows the user to select among a Day View 307, Week View 308-309, or Month View 310 of his or her personal calendar.

Date entry field 1104 allows the user to type in a date, and by clicking the Go button 1105, go to the My Calendar screen 307 for the indicated date. Options button 1106 takes the user to a screen (not shown) where preferences and personal options can be set. Help button 805 is a link to Help pages 326, and Log Out button 806 exits the system.

Add Appointment button 1101 allows a user to manually add an event or appointment to the personal calendar, using a screen similar to that shown in FIG. 14, described below. In this way, the invention can be used as a tool for integrating personal events and appointments in a calendar with selected events obtained from an outside source, by overlaying the two types of events as layers in a single calendar. In one embodiment, the user may also click on buttons 1102 to add an appointment at a particular time of day.

In an alternative embodiment, My Calendar page 307 may also include a list of "to-do" and/or "all day" items for the day. These items generally do not have a specific time of day associated with them; rather, they are items that the user has entered to remind him or her of tasks to be performed sometime during the day. The user can add such items using the Add Appointment button 1101, in much the same way as appointments are added.

Also shown is a daily planner 1110 listing events and appointments 1112 for various times throughout the day. These events and appointments are obtained by reference to the user's personal records in database 112 reflecting a manually entered item (such as a lunch appointment), or they may be descriptive of events obtained from events database 114 representing event data from an outside source (such as, for example, a concert or movie). In one embodiment, events selected by the user from an outside source are replicated in the user's personal records in database 112, so that he or she can freely modify them. In another embodiment, events selected by the user from an

outside source are not replicated, but rather a reference (or "pointer") to the event in database 114 is stored in the user's personal records, allowing the invention to access the data describing the event from the outside source. One advantage to the latter technique is that any subsequent updates to the event can automatically be reflected by the application when the user views the My Calendar screen.

In one embodiment, the user can click on any event shown in list 1110 or in the list of "all day" events 1111, to be presented with a screen showing more detail on the selected event. The user may also edit the event if desired, as described below in connection with FIG. 14.

In another embodiment, a link may be provided for making a purchase associated with a particular event. For example, if the event is a concert, a link to an on-line ticketing service maybe provided, for purchasing tickets to the concert. The link can be targeted to a particular event within an electronic commerce site that sells tickets, so that the user need not re-enter the particulars of the event in order to purchase tickets. Alternatively, such an approach may be used to register for an event such as a tradeshow, by providing a link to a web page allowing the user to enter registration information, payment information, and the like. Where possible, certain fields of the web page may be pre-filled with default information based on a profile previously entered by the user and stored in database 112.

Similar techniques can be used for on-line sale of books, music, and the like, in connection with calendared events such as release dates. In one embodiment, the user may use the present invention to track birthdays, anniversaries, and the like; the system may then remind the user of such an event and provide the opportunity to buy cards, gifts, and the like, as well as provide direct links to electronic commerce pages and sites that are of interest in connection with the upcoming event. The system of the present invention can thus facilitate on-line commerce related to events that are of particular interest to the user.

In one embodiment, a subscreen 1109 may be shown when selected by the user, displaying a list of Favorite Events. Subscreen 1109 contains a section for each event category selected by the user. Individual events 1113 associated with each event category, and belonging to the current date, are displayed. The user can click a button 1114 to add an event 1113 to the personal calendar for display on the My Calendar screen 307.

In one embodiment, the user can select which categories of events are to be displayed in the My Calendar screen 307, so that he or she can view a subset of subscribed categories, if desired. This provides a multi-layering function, wherein each category can be considered a "layer" that can be viewed in combination with other layers as selected by the user. The user can also select group calendars for display; events from such calendars are shown in the My Calendar screen as an additional "layer", if desired.

Referring now to FIG. 12, there is shown a screen shot of a Week View of a My Calendar screen 308 according to one embodiment of the present invention. Screen 308 operates in a similar manner as the Day View 307 described above in connection with FIG. 11. Events for a week-long period of time are shown, in list format 1202. Grid button 1205 provides access to a screen (not shown) that displays the week's events in a grid format. Links 1203 provide access to individual Day View screens for each day in the week. Buttons 1204 open a Favorite Events tracker subscreen similar to 1109 in FIG. 11, for displaying individual events within categories. Events 1112 are shown as links, which when activated access a detail screen for each event. Buttons

15

1101, 1102, 1106, 805, 806, and 1105, and field 1104 operate in a manner similar to the corresponding buttons in FIG. 11.

Referring now to FIG. 13, there is shown a screen shot of a Month View of a My Calendar screen 310 according to one embodiment of the present invention. Screen 310 operates in a similar manner as the Day View 307 described above in connection with FIG. 11. Events for a month-long period of time are shown, in grid format 1302. Links 1203 provide access to individual Day View screens 307 for each day in the month. Events 1112 are shown as links, which when activated access a detail screen for each event. Buttons 1101, 1106, 805, 806, and 1105 operate in a manner similar to the corresponding buttons in FIG. 11.

Referring now to FIG. 14, there is shown a screen shot showing a My Calendar detail screen 311, containing detailed information describing an event. This subscreen can be used for entering information for a new event, or for making changes to information for a previously entered event. It is accessed by clicking on an event 1112 in screens 307, 308, or 310.

Event description 1401 contains several fields for displaying and making changes to various pieces of information concerning the event, such as a title 1402, date 1403, start time 1404, duration 1405, check box 1406 for all-day events, and recurring event information 1408, 1409, and 1410. A notes field 1407 is also provided for miscellaneous information. Fields 1402 to 1410 are populated with information from database 112 for manually-entered appointments, or with information from database 112 or 114 for other events. In alternative embodiments, other fields may also be provided, such as a reminder field for gift-related events, and the like. The selection of particular additional fields to be provided may be customizable.

In an alternative embodiment, screen 311 contains a link to another page allowing the user to perform other activities related to the scheduled appointment (such as an online purchase relating to the item).

Once the information has been entered, edited, and/or verified, the user can click on the Save Appointment button 1411 to save the entered information and return to the My Calendar screen. Alternatively, the user can click the Back button 1412 to cancel all changes made to the event and return to the My Calendar screen. Finally, the user can delete the appointment by clicking button 1413.

#### Application Implementation

Referring again to FIGS. 1 and 1B, the operation of application server 106 to implement the application will now be described. Generally, application server 106 responds to requests relayed by web servers layer 102. Generally, such requests are provided to server 106 in HTTP format. In response to such a request, server 106 optionally performs some action (if appropriate) and returns an HTML page to the user. In one embodiment, HTTP requests are relayed to a load balancer 105. Load balancer 105 dispatches the request to the appropriate process 108.

In one embodiment, each request is handled by a class in an object-oriented language such as C++. Server 106 inspects its registry to determine which class to invoke for each request. The Uniform Resource Locators (URLs) requests to server 106 contain the name or identification of the class that can handle the request.

To handle a request, server 106 creates an instance of the appropriate class and invokes its "Execute" method. The parameters of the HTTP request are made available to the class as a name-value pair list in the base class. The Execute method parses the input parameters, performs actions, and generates HTML output.

16

Referring now to FIG. 16, there is shown a flowchart of the basic operation of the Execute method. The method performs the following actions:

1601: Calls the class's Init method. The default Init method provided by the base class handles user authentication 1601A, setup of sessions 1601B, and binding 1601C of common resources needed by the class.

1602: Calls the class's DoExecute method. The DoExecute method performs the actual work and returns HTML data. The actual DoExecute method to be performed is determined by the particular subclass being implemented, which typically overwrites the default DoExecute method.

1603: Calls the class's Finalize method. The default Finalize method releases 1603A session and default resources.

The DoExecute method for a particular class generally overrides the default DoExecute method 1602, and performs one or more of the following tasks:

Parse parameters included in the request into members of the class or temporary variables, as needed.

Perform some action by calling lower level service Application Programming Interfaces (APIs); for example, saving data to a database.

Output a componentized HTML document. This may be implemented, for example, by calling the template engine in conjunction with a Parts Map, as described in more detail below, along with user interface "parts" and related helper classes to construct an HTML page.

The base class provides general purpose Init 1601 and Finalize 1603 methods for most classes. Classes that require login are automatically authenticated by the base class, freeing them from having to implement authentication redirect logic. The base class performs parameter checking to provide security by detecting invalid requests. The base class also performs logout operations when needed. In addition, the base class provides several general-purpose methods for handling, for example, building of URLs for redirection, saving temporary session data, and parsing simple parameters.

Referring now to FIG. 17, there is shown a flowchart of the user authentication process 1601A as handled by the base class. Classes that do not require authentication may circumvent this process by overriding a RequiresLogin method to return FALSE.

Two separate sessions are maintained for each user, in connection with the authentication process of FIG. 17:

a long-term session, managed by the load balancer and assigned a session ID (this session is automatically bound to future requests using browser cookies, as is known in the art); and

a calendar session, managed by a session manager (created upon authentication with the calendar service, assigned the same ID as the long-term session).

The base class first checks 1701 for an existing long-term session. If none exists, one is created 1702. The base class then retrieves 1703 the Session Id and looks up 1704 a calendar session having that ID. If a calendar session having the corresponding ID exists 1705, the user is authenticated 1706 and the authentication is checked 1707 to see if it is valid. If so, the user is allowed to proceed; he or she is redirected 1709 to his or her default calendar view and the authentication process ends. If the authentication is invalid, access is denied 1708.

If in 1705 no calendar session having the corresponding ID exists, the base class checks 1710 for an "auto-login"



cookie on the user's machine. Auto-login allows users to return to the web site without having to login. The auto-login cookie contains the user's encrypted login name and password. If the cookie exists, the base class retrieves 1711 the login information. If the cookie does not exist, the user is redirected 1712 to a login screen. In one embodiment, the base class saves the original URL that the user was trying to access. The user is then prompted 1713 to enter login information.

Login information from the auto-login cookie or from the user's entry is checked 1714 for validity. If it is valid, the base class creates 1715 a calendar session for the user, and the user is allowed to proceed. If a URL was previously saved (in 1712) the user is redirected 1709 to that location; if not, the user is redirected 1709 to his or her default calendar. If in 1714 the login information is invalid, the user is denied access 1716.

In addition, in one embodiment a user may be automatically logged out by the session manager if they remain idle for some predefined period of time, or when the server is heavily loaded. In this case, the user must login again when he or she returns.

Referring now to FIG. 18, there is shown a block diagram of the operation of a template processor 1804 to generate an HTML file. This technique is used in one embodiment of application server 106, with the template processor operating as part of a process 108 to generate output for presentation to the user. Template source file 1801 contains HTML and process tags. Processor 1804 uses process tags to determine what items in file 1801 need to be replaced with dynamic data.

Template map object instance 1802 is used by processor 1804 to implement simple replacement of identifiers in source file 1801. Map 1802 contains a list of name-value pairs. For each process tag in file 1801, processor 1804 consults map 1802 for the name specified in the tag. If the name is found, the value associated with the name is inserted in place of the process tag.

Template data object instance 1803 is used by processor 1804 to fill in data elements in a template file. For example, object instance 1803 may contain a list of items with attributes, such as events, calendars, and the like. Processor 1804 uses object instance 1803 to iterate through a set of data, replacing process tags with the attributes of the elements.

Thus, to process a template source file 1801, processor 1804 scans through file 1801 for process tags. When it finds such a tag it requests values from template map object instance 1802 and template data object instance 1803. These values are used to replace the process tags. Processor 1804 also uses template data object instance 1803 to iterate over data sets referenced by process tags.

For example, consider a source file 1801 containing the following source:

```
<process type=cell id=UserName>ReplaceMe</process>
<process type=tile id=events>
  <process type=cell id=events.title>Event
  Title</process>
  <process type=cell
  id=events.date>12/31/1999</process>
</process>
```

Processor 1804 would scan through the file and first find the UserName cell tag. The type cell indicates that a simple replacement should be done, with data from either the template map 1802 or the template data object 1803. Tem-

plate map 1802 is checked first for a UserName element. If no value is found there, template data object 1803 is checked. If a value is found, the text "ReplaceMe" is replaced with the value. If no value is found the text "ReplaceMe" remains intact.

Processor 1804 would then find the process tag with the keyword events. The type tile indicates that processor 1804 should loop over a data set contained in template data object 1803. The keyword events identifies which data set to use. Processor 1804 calls the following methods of the template data object 1803:

IsEmpty: Determines if the data set events is present in template data object 1803;

MoveNext: Moves to each element in the events data set, returns FALSE when there are no more elements;

GetValue: Retrieves individual attributes from the current data element in events.

Thus, in the example given above, processor 1804 would call MoveNext for as many events included in the events data set. Processor 1804 would then call GetValue to obtain the events. title and the events. date attributes for each element in the data set.

Once processor 1804 has processed all of the process tags in source file 1801, it generates an HTML file 1805 which can be passed to the user and read by a browser.

Operation of processor 1804 will now be described in more detail. Referring now to FIG. 19, there is shown an example of a document template 1900 containing several parts, including header part 1901, tabs part 1902, navigation part 1903, day grid part 1904, event tracker part 1905, and footer part 1906. Parts 1901-1906 are modular user interface components used to implement and generate the various pages for the present invention. Thus, an HTML document is modularized into parts that can each be generated by a distinct class 2001 run by application server 106.

Referring also to FIG. 20, there is shown a block diagram of the detailed operation of template processor 1804 to generate HTML output 1805.

To implement the component model exemplified by document template 1900, two levels of template evaluation are used:

Document-level template evaluation (controlled by the class) specifies which user interface components (parts) are included in a particular document, and how they are to be laid out. A document-level template 1900 is used for this purpose. In addition, the template may include some common formatting and style elements.

Part-level template evaluation (controlled by the parts within a document) is used to evaluate each part in the document by consulting a part template file 2006. The HTML data 1805 is then generated for that part's particular area in the document.

For each document, a single document-level template evaluation is performed, and any number of part-level template evaluations are performed.

In the document-level template 1900, custom process tags are employed to specify parts to be included in the document. For example, the following example contains three parts:

```
<process type=whenHeader title="My Calendar Day
View">Replace Text</process>
<process type=whenTab template=
myCalendarTab.html>Default Text</process>
<process type=whenFooter>Replace Text</process>
```

Part map 2002 translates each part tag into a section of HTML. Part map 2002 is passed by classes 2001 to

template engine 2005 when document-level template evaluation takes place. Parts map 2002 acts as a dispatcher, interpreting custom tags and supplying needed parts classes from parts library 2003. When engine 2005 requests that parts map 2002 translate a custom tag, map 2002 dispatches control to an instance of the appropriate part class from parts library 2003 by calling its Output method.

The Output method of each part generates the HTML that represents its user interface component in the document. Template engine 2005 replaces the custom process tag with the HTML data generated by the Output method. Most parts use the template engine 2005 to create their HTML output, in order to implement part-level template evaluation. The part-level evaluations actually cause re-entrance of template engine 2005 since the part evaluates its template 2006 during a callback from document-level template processing.

Template class 2001 contains the IsEmpty(), MoveNext(), and GetValue() methods described earlier.

Thus, for example, in implementing a calendar-oriented parts, the following elements are employed:

Part template file 2006: contains the HTML and process tags for creating the part output.

Part class implementation in template class 2001: A C++ class that implements the Output method. This class embodies the presentation logic of the part, and may defer to helper classes for core presentation algorithms.

The template class 2001 may perform any of the following, for example:

Establishing references to the cache of user data from the session manager.

Establishing a context object for communicating with the calendar engine.

Finding the data needed during template evaluation (including, for example, accessing the calendar engine to obtain lists of events for all the days being displayed).

Parsing input parameters received with the HTTP request into arguments used during template evaluation (such as, for example, start date, end date, calendar ID, event IDs, and the like).

Optionally creating a Template Map class and populating it with name-value pairs used for replacement during template evaluation.

Invoking template engine 2005, passing it the template file name, template map, and custom template data object.

The template class 2001 objects may perform any of the following, for example:

Provide convenient constructors for evaluation by the part. The constructor accepts arguments needed during template evaluation. The constructor may also validate data and transform it into a more usable form.

Override the IsEmpty() method for the data set that the template data serves. A single template data class object may serve more than one data set. For example, the Day Grid part's Template Data class serves two data sets: a collection of all-day events and a collection time intervals. IsEmpty() is called by template engine 2005 when it begins processing a data set. IsEmpty() determines whether there is any data in the set, and if so, position itself to point to the first element in the data set.

Override the MoveNext() method to move to the next available element in the data set. If there are no more elements, MoveNext() returns FALSE. Template class

2001 may implement MoveNext() by incrementing an internal counter or by advancing the position of an event list obtained from the Calendar Engine.

Override the GetValue() method to return values for each attribute of the current data element. This may include accessing methods of calendar engine objects, formatting them, and returning them in a buffer. GetValue() is also used to show or hide areas in a template. By returning a null string, GetValue() can effectively hide a section of the template. This may be used to implement conditional template evaluation.

Classes 2001 and Parts 2002 access the data they require prior to starting template evaluation. Generally, such data includes lists of events obtained from calendars. Other types of data that may be required include, for example, user profiles, calendar attributes, lists of tracked calendars, weather information, and the like. The calendar service serves up most of the data required by the user interface presentation logic.

#### Session Management

As described above in connection with FIG. 1B, application implementation element 121 includes session management 123 which authenticates, tracks, and manages user sessions for calendaring operations, and stores a cache of user and calendar data as needed to maintain sessions. The operation of session management 123 will now be described in more detail.

Referring again to FIGS. 1A and 1B, when a user is authenticated, a session is created, as is known in the art of web application development. The session exists until either the user logs out or the session expires. In one embodiment, a session expires when the user remains idle for a certain time period as specified in system configuration files.

Sessions serve two primary purposes: 1) they obviate the need for reauthentication of the user with each web page access; and 2) they facilitate the use of a cache to store data (such as calendar data) in order to avoid unnecessarily retrieving the same data repeatedly from databases 112 or 114. In one embodiment, only one thread 115 of a process 108 may access a session object at any given time. Other requests to access a session object are blocked until the thread 115 that is currently accessing the object is done. This technique reduces the amount of thread safe checks needed to assure data integrity.

When a user logs on and a session commences, certain information may be pre-fetched from database 112 and transferred to user cache 109, so that subsequent access to this data will be faster. The pre-fetching occurs immediately after login and, preferably, prior to the user accessing his or her calendar. The pre-fetching need not be complete for the user to access his or her calendar.

Periodically, session management 123 prunes the list of stale sessions by removing sessions that have been idle for longer than a specified time-out period, specified in pre-defined configuration information. In addition, in periods of high load session management 123 may remove a session before the time-out period is reached, as further specified in the configuration information. A user attempting to access the system after his or her session has been removed must logon once again to re-establish a session.

#### Calendar Service 124

As described above in connection with FIG. 1B, application server 106 includes calendar service 124 which implements the core object model for accessing user profiles, calendars, events, calendar links, and collections of objects; implements event lookup, object caching, and persistence; and serves both personal calendars and event directory data

using a common object model. The operation of calendar service 124 to implement the present invention will now be described in more detail.

Referring now to FIG. 21, there is shown a block diagram of calendar service 124 according to one embodiment of the present invention. Calendar service interface 2101 defines the contract between calendar service 124 and the application servers layer 103 of system 100, or of any other client of service 124. Core calendar engine 2102 implements the core algorithms and logic of the calendar engine. Data access layer 2103 provides the persistence behavior of all objects in calendar service 124. It implements access to both relational data 2104 (user data) and memory resident data 2105 (event directory).

Calendar Service Interface 2101. Calendar service 124 provides the programming interface for accessing objects. In one embodiment, these objects are the only interface to calendar service data that is available to application servers layer 103 or any other client of service 124. These objects include, for example:

User Account: Contains a user's profile, personal calendar, referenced calendars, and display preferences. The user account is the primary object stored in the cached session object.

Calendar: Provides the common interface to calendars of all types (both personal and event directory schedules). Calendars are containers for individual events, and they interact with Event List and Day List objects to provide access to the events they contain.

Event: Provides a common interface to events of all types, including personal events, event directory events, and linked events. The same class is used to represent recurring and non-recurring events.

Calendar Link: Represents some relationship between a user and a calendar. For example, there may be a "tracking" relationship for the event tracker, as described above. A calendar link provides access to the underlying calendar object instance.

Day Link: Provides a convenient collection of events partitioned by day. The day list contains a set of Event List object instances—one for each day within a certain date range. Can also be used as an iterator to traverse through the contained Event Lists one at a time.

Event List: Provides a collection of events sorted in various ways (such as by date, title, and the like). Can also be used as an iterator to traverse through the contained events one at a time.

Calendar List: Represents a collection of "linked" calendars. Contains a set of Calendar Link objects. Each Calendar Link points to a Calendar object instance. The Calendar List can be sorted by various means.

Sponsor/Category/Partner: Provides the interface to both sponsor and category information. A calendar can have an associated sponsor as well as an associated category and an associated partner, if desired.

Object Ownership Hierarchy 2200. Referring now to FIG. 22, there is shown an example of an object ownership hierarchy 2200. Calendar service 124 provides a global calendar manager object 2201. Calendar manager object 2201 is the starting point for accessing objects in calendar service 124. In one embodiment, all calendar service objects are indirectly created through calendar manager object 2201.

Calendar service 124 employs an object ownership hierarchy to control the life cycles of objects. For example, event objects are owned by a single calendar object. The calendar object acts as a container for each event, and also controls the access rights, modification, deletion, and creation of the event.

In FIG. 22, calendar manager 2201 serves up both user account objects 2202 and public calendar objects 2206 (event directory). For user accounts, authentication is required before access to a user account is granted. This can occur either via user login or by creation of a new user account, as described previously. Once a user account object 2202 has been created, it is possible to access the user's personal calendar object 2204, tracked calendar links 2203, and personal profile through the user account object 2202 instance.

For public calendars, authentication is not required. Calendar manager 2201 searches for the desired calendar and returns an object instance that provides access to the calendar's events and attributes 2206 and 2207.

Access Control. Access to calendar service objects and methods is controlled in two ways:

Explicit access checking: Certain operations require that the caller's identity be explicitly compared with the access permissions of the object being requested. For example, calendar manager 2201 requires a valid login name and password before it will create a user account instance 2202.

Implicit access: Access to some objects is controlled indirectly by relying on some previous explicit access check. For example, once access to a user account has been granted, the caller is free to access the contents of the associated personal calendar and calendar links without further access checks. This is implemented using session management, as described previously.

Access control is implemented using a context object. Calendar manager 2201 creates context objects for callers of calendar service 124. Operations that require an explicit access check are defined to take a context parameter. The context identifies the user performing the operation and the access level of that user. Calendar objects check the context permissions against the operation being performed to ensure appropriate access. In this way, security leaks can be avoided even when the application code contains a programming bug.

Object Creation and Modification. Objects are created and modified by their owning parent object, according to the defined hierarchy. Internally, objects implement their own interfaces.

Parent objects control the modification, deletion, and creation of objects they contain. This allows the parent to modify its internal structures when its contents change. Furthermore, the implementation class can be hidden from clients of calendar service 124, so that parent objects act as an object factory for objects they contain.

Creating a new object involves two steps. First, the caller requests a new instance of an object type from the parent. The parent creates a new modifiable object instance and returns it to the caller. The object instance does not yet persist in the calendar service database. Second, the caller requests an "update" of the contained object from the parent. The object may be an existing object or a new object. The caller implements the update by writing changes to the database and updating its internal structures. For example, when a calendar updates a recurring event, it may need to recreate the expanded event instances due to a change in the recurrence rule.

When deleting an object, the caller merely asks the parent to delete the contained object.

Releasing Objects. As described previously, parent objects provide the means to access instances of their contained objects. Parent objects also control the release of their contained objects. When a client of service 124 is

finished with a calendar service object, it notifies the parent, which releases the object instance.

This allows the implementation of calendar objects to be hidden from clients, so that the constructors and destructors of object classes are restricted (protected). Thus, the parent object can implement object memory and resource management in any manner desired.

**Calendar Types and Domains.** In one embodiment, calendar service 124 provides a single interface class for calendars. This interface class is used for both personal calendars and event directory schedules (public calendars) even though they have different attributes and persistence models. Calendar service 124 provides the concept of a "domain" for any object in the system. This domain is used, for example, to distinguish between personal and public calendars and events.

**Accessing Calendar Contents:** Calendar service 124 provides two types of access to events in calendars. Each access type has several variations, to support the needs of application servers layer 103.

The first type of event lookup serves a series of events in a Day List. A Day List is a list of event lists. Each list contains a set of events for a single day. The Day List contains an event list for each day in its date range (for example, a week of events would be sorted into seven event lists). The Day List lookup is convenient for displaying calendar-oriented views of events, since it is very easy to determine the number of events on any given day within the date range. The Day List lookup interface is implemented by a Day List object. Callers use a `SetTimeSpan()` method to generate the list of event lists from either a calendar or a list of calendars.

The second type of event lookup serves events for a specified date range in one contiguous event list object. The Event List lookup is used to display lists of events (such as for pages 307 and 308) rather than calendar-oriented displays (such as pages 309 and 310). The Event List lookup interface is implemented by a Calendar class. Callers use a `GetEventList()` method to obtain an aggregate event list. Variations of `GetEventList()` include:

**Date Range:** Returns all events from a calendar within a specified date range.

**Page Number:** Returns a certain page of events starting from a specified date. A page is defined to be a certain number of events. For example, if the page size is 50, then page 2 would return an event list containing events 51 through 100 starting on the specified date.

**Layered Calendar Views.** Layered Calendar Views display the contents of multiple calendars on a single calendar display such as Views 307, 308, 309, or 310. Layered Calendar Views are implemented by calendar service 124 as a Day List object that contains events from multiple calendars. More specifically, the Event Lists contained by the Day List contain events from multiple calendars.

The Day List class implements the Layered Calendar View. Callers use a `SetTimeSpan()` method of the Day List and pass it a calendar list specifying all the calendars to be included in the display.

**Core Calendar Engine 2102.** Core calendar engine 2102 implements the core algorithms and logic of calendar service 124, including:

- Object creation, deletion, and modification;
- Control of object persistence operations;
- Event lookup for day lists and event lists;
- Object collections and sorting;
- Event caching;

Time-zone translation of events; and

Expansion of recurring event instances from recurrence rules.

**Interface and Implementation Classes.** As described previously, calendar service interface 2101 defines a set of classes that define the contract between application servers layer 103 and calendar service 124. Core calendar engine 2102 returns instances of these classes to the caller.

In one embodiment, the implementation of these objects is achieved by calendar service 124 defining a set of "engine" subclasses that embody the implementation. The implementation classes are collectively referred to as the CE Layer classes, while the interface classes are known as the CI Layer classes.

**Calendar Manager 2201.** Calendar Manager 2201 is a global object, created at service startup. It creates and manages user accounts, public calendars, partner objects, and context objects.

Many calendar service operations use a context parameter to validate the access rights of the caller. In one embodiment, a context object is employed to embody access information. The context object includes, for example:

The User ID of the person initiating the request (a User ID of zero indicates guest access).

Access Mode: Provides information as to the access rights of the user.

Reference to the template class making the request.

Reference to a database transaction object.

Contexts can be created as follows:

**CreateGuestContext** returns a context object with guest access rights. The guest rights are not associated with any particular user.

**LoginNewUser** creates a context object as part of new account creation. The context has the access rights of the new user account that is created.

**LoginUser** creates a context object as part of the login process. The context object has the access rights of the user account that is authenticated.

A context object can be used for any number of calendar service 124 operations. In one embodiment, state information is not maintained in the context. Session management 123 caches a context object in user session data for all calendar service operations during the life of the user session.

**User Account Login and Registration.** As described previously, the `LoginNewUser` and `LoginUser` methods control user registration and login respectively. `LoginNewUser` creates a new user account implementation object (`CDUserAccount` class), fills in the profile information, and saves the account object to the user database. The account object creates itself and a set of other objects associated with the user account (personal calendar object, calendar links list).

Calendar manager 2201 provides access to the event directory calendars via the `GetCalendar` method. `GetCalendar` takes a calendar ID and returns a calendar object. In one embodiment, event directory calendars are obtained from calendar manager 2201, while personal calendars are accessed from a user account object.

To implement public calendar lookup, calendar manager 2201 creates an instance of a public calendar (class `CDPublicCalendar`), initializes it with the requested calendar ID, and instructs the calendar object to load itself from the database. The `CDPublicCalendar` class implements the database lookup as described below.

On return from `GetCalendar`, the caller has a pointer to a calendar interface object (class `CICalendar`). Typically, the

caller releases public calendar objects once the template class request has been processed. This is the case for Event Directory pages. However, Public Calendar instances are used to represent a user's tracked calendars. In this case, the Public Calendar instances are cached by the user account object—one for each tracked calendar.

Calendar manager 2201 also serves partner objects. The partner object is a denormalized object containing information about categories and sponsors. Category/sponsor information is referenced by calendars in the event directory. Attributes of the category/sponsor are used in the various pages of the user interface (for example, sponsor name, logo, URL, and the like).

At startup, calendar manager 2201 loads all partner objects from the database and caches them in memory. Calendar manager 2201 provides a single call to get a partner object by ID. Public calendars use this call to return their associated partner object.

User Account Implementation. In one embodiment, user accounts 2202 contain all information associated with a particular user within calendar service 124. User account 2202 is the attachment point for all user data and is the primary object stored in the session cache 109. User account 2202 provides access to the user's personal calendar, the linked calendars (favorite events), and the user's profile information.

User account 2202 provides sole access to a user's personal calendar. Personal calendar object 2204 is created by user account 2202 at registration or login. Personal calendar 2204 has no attributes of its own and is therefore not actually stored in the database. Rather, personal calendar 2204 is an instance of a calendar object that manages a set of events.

Calendar link 2203 defines a relationship between user account 2202 and a calendar. The relationship can have a type, such as a "tracked calendar" relationship. Calendar links 2203 can be used to denote other types of relationships between users and calendars as well.

User account 2202 manages the set of all calendar links 2203. It controls the creation, modification, and deletion of link objects, and controls their persistence operations, but in one embodiment does not implement them. When a user account 2202 is loaded, it performs a query for all calendar links 2203 and loads them into an internal list. User account 2202 provides an interface to obtain a copy of the list of calendar links 2203. The list can be sorted in any of a number of ways. Each list copy references the same object instances.

The attributes of a user account 2202 fall into two categories: profile attributes and application settings. Profile attributes define the user's demographics and core account information. The user account 2202 provides methods to change profile attributes. These changes are made to a modifiable instance copy of the user account 2202 object, obtained from calendar manager 2201. Changes to profile attributes are saved immediately in the database.

Application settings are fields that control the behavior and appearance of the application. These may include, for example:

- the order of events in the event tracker;
- the expand/collapse state of categories in the event tracker; and
- the minimize state of the event tracker.

Some application settings are saved immediately when changed, while others are saved when the user's session expires through either explicit logout or session time-out.

The user account implementation class provides a protected method (FlushDeferredChanges) which calendar

manager 2201 uses to save application settings prior to deleting the user account object. Deferred write is used in order to minimize database accesses for common application customizations.

Calendar Implementation. In one embodiment, calendar service 124 supports two types of calendars: 1) personal calendars, which contain user-specific private data loaded directly from database 112; and 2) public calendars, which represent schedules in the event directory database 114 which are accessible by any user and whose data is referenced from the shared memory event cache 110.

In one embodiment, the difference between the two types of calendars is the persistence mechanism used to inflate and deflate the calendar and event objects from the two domains.

Before events are served up by the calendar, they are loaded into memory as a set of event object instances. Calendar service 124 maintains an internal event list collection object containing object instances for all event occurrences whose dates fall within the loaded date range, including each instance of a recurring event that falls within the loaded date range. The cached event instances are also translated to the calendar's target time zone and sorted by time.

A calendar implementation class controls the loading and unloading of events over the loaded date range. The calendar implementation class implements a simple loading model which enforces a contiguous range for this date range. The maximum size of the date range is specified in server configuration files, for example as 50 days. The minimum load range size is also specified in server configuration files, for example as 10 days. The minimum load range size defines the "chunk" size for database queries.

Cache 110 serves to optimize performance by minimizing database accesses to personal calendar. Furthermore, cache 100 pre-sorts events by date and time for the target time zone, as will be described below.

Each calendar has the notion of a "zone focus." The zone focus is set by the client of calendar service 124 based on the time zone of the user's display. For event directory pages, the zone focus is set to the default time zone for the calendar being viewed. The zone focus for personal calendars is the user's time zone as specified in his or her profile. This is also true for "tracked" public calendars (calendars which are displayed with or over the personal calendar).

When a calendar loads its events it also translates the events to its zone focus. This is done so that the calendar's internal event list can be presorted based on the target time zone. Event list sorting is done when the target time zone is known. All-day events do not shift across date boundaries as do events having a specific start time. Pre-sorting the internal cache by date allows event lookups to be highly optimized, requiring no list duplication or sorting.

Recurring Events. A pattern of recurring events can be stored in one database row in database 112 or 114. The individual occurrences of the recurring pattern need not be stored individually. Recurring patterns are stored in a table separately from non-recurring events.

Calendar service 124 loads all recurring patterns from the database the first time the calendar is loaded. The recurring patterns are saved as event object instances in a separate internal recurring event list. When calendar service 124 loads events, it also generates individual event instances from the list of recurring patterns. Thus, calendar service 124 creates an event object instance for every occurrence of each recurring pattern whose date falls in the loaded date range. These recurring event instances are placed in the internal cache in the same manner as non-recurring events.

27

Each recurring event instance is assigned a unique ID which is a combination of: 1) the event ID of the recurring pattern from which the instance was generated; and 2) the Julian date assigned to the recurring instance. The combined ID is used to uniquely identify recurring event instances by clients of calendar service 124.

When the user modifies the pattern of a recurring event (for example, changes from repeat daily to repeat weekly), calendar service 124 performs the following steps:

- 1) Delete all the existing recurring event instances associated with that pattern from the internal cache;
- 2) Create new recurring event instances in the internal cache based on the modified pattern; and
- 3) Save the recurring pattern to the database.

Furthermore, calendar service 124 is equipped to handle a change from a recurring event to a non-recurring event, or vice versa. For example, if a user wishes to change a recurring event to a non-recurring event, calendar service 124 performs the following steps:

- 1) Delete all recurring event instances associated with that pattern from the internal cache;
- 2) Delete the recurring pattern "master" event from the internal pattern cache;
- 3) Delete the database row associated with the "master" event pattern;
- 4) Insert the non-recurring database row for the event; and
- 5) Create and possibly insert the new event object instance into the internal cache.

A similar but opposite set of actions is performed when converting a non-recurring event to a recurring event.

From the above description, it will be apparent that the invention disclosed herein provides a novel and advantageous system and method of multi-layered online calendaring and purchasing. The foregoing discussion discloses and describes merely exemplary methods and embodiments of the present invention. As will be understood by those familiar with the art, the invention may be embodied in other specific forms without departing from the spirit or essential characteristics thereof. Accordingly, the disclosure of the present invention is intended to be illustrative, but not limiting, of the scope of the invention, which is set forth in the following claims.

What is claimed is:

1. A computer-implemented method for multi-layered online calendaring, comprising:

- a) accepting user input specifying at least one of a plurality of categories; wherein at least one category of the plurality of categories corresponds to a unique visual characteristic;
- b) retrieving a plurality of events associated with the specified category, each having a date;
- c) selectively displaying a date-delimited subset of the retrieved events associated with the specified category;
- d) accepting user input selecting at least one of the displayed events;
- e) adding the selected event to a computer-implemented personal calendar;
- f) repeating steps a) through e); and
- g) displaying the computer-implemented personal calendar by overlaying information from the selected events from the specified categories on a common display, the information for at least one event in a category being displayed uses the unique visual characteristic corresponding to the category if the category corresponds to a unique visual characteristic.

28

2. The computer-implemented method of claim 1, wherein each of at least a subset of the events has a time.

3. The computer-implemented method of claim 1, wherein the unique visual characteristic is a color.

4. The computer-implemented method of claim 1, further comprising:

- f) repeating steps a) through e);
- g) accepting user input specifying a subset of the specified categories for display; and
- h) displaying the computer-implemented personal calendar by overlaying information from the selected events from the subset of the specified categories on a common display.

5. The computer-implemented method of claim 1, further comprising:

- d.1) accepting user input specifying a purchase associated with the selected event; and
- d.2) processing the specified purchase.

6. The computer-implemented method of claim 1, further comprising:

- f) displaying the computer-implemented personal calendar.

7. The computer-implemented method of claim 1, further comprising:

- f) displaying a date-delimited subset of the computer-implemented personal calendar.

8. The computer-implemented method of claim 1, further comprising:

- f) accepting user input describing at least one personal event; and
- g) integrating the personal event with the selected event in the computer-implemented personal calendar.

9. The computer-implemented method of claim 8, further comprising:

- h) selectively publishing the personal event to a group calendar.

10. The computer-implemented method of claim 9, further comprising:

- i) accepting input specifying a group of users to have access to the group calendar;
- j) accepting input from a second user requesting information from the group calendar; and
- k) responsive to the second user being specified in the group, selectively transmitting the information from the group calendar to the second user.

11. The computer-implemented method of claim 9, further comprising:

- i) accepting input specifying a group of users to have access to the group calendar;
- j) designating a permitted level of access for each user in the group;
- k) accepting input from a second user requesting access to the group calendar;
- l) responsive to the second user being specified in the group, and to the requested access corresponding to the permitted level of access for the second user, allowing the requested access to the group calendar; and
- m) responsive to the second user not being specified in the group, or to the requested access not corresponding to the permitted level of access for the second user, denying the requested access to the group calendar.

12. The computer-implemented method of claim 1, further comprising:

- f) accepting user input describing at least one personal event; and

29

g) displaying the computer-implemented personal calendar by overlaying the personal event onto the at least one selected event from the at least one specified category on a common display.

13. The computer-implemented method of claim 1, further comprising:

- f) accepting user input modifying the selected event;
- g) recording the modified selected event in the computer-implemented personal calendar.

14. The computer-implemented method of claim 1, wherein a) comprises:

- a.1) transmitting a page to a screen viewable by the user, the page containing a plurality of links, each link associated with a category; and

- a.2) detecting user activation of a link specifying one of the categories;

and wherein c) comprises:

- c.1) transmitting a page to a screen viewable by the user, the page containing a plurality of links, each link associated with a retrieved event belonging to a date-delimited subset of the retrieved events associated with the specified category;

and wherein d) comprises:

- a.2) detecting user activation of a link specifying one of the displayed events.

15. The computer-implemented method of claim 14, wherein each of steps a.1) and c.1) comprise transmitting a page via a hypertext transfer protocol, and wherein each transmitted page comprises a hypertext-encoded document.

16. The computer-implemented method of claim 1, wherein each of steps a) and c) are performed by transmitting a file across a computer network.

17. The computer-implemented method of claim 1, wherein b) comprises retrieving a plurality of events from an externally-hosted database.

18. The computer-implemented method of claim 1, further comprising:

- f) accepting user input specifying that the computer-implemented personal calendar is to be shared with a second user; and
- g) accepting input from the second user requesting the computer-implemented personal calendar; and
- h) selectively transmitting at least a portion of the computer-implemented personal calendar to the second user.

19. The computer-implemented method of claim 1, wherein c) comprises selectively displaying a subset of the retrieved events associated with a user-specified date.

20. The computer-implemented method of claim 1, wherein c) comprises selectively displaying a subset of the retrieved events associated with a user-specified month.

21. The computer-implemented method of claim 1, wherein c) comprises selectively displaying a subset of the retrieved events associated with a user-specified week.

22. The computer-implemented method of claim 1, wherein b) further comprises:

- b.1) retrieving at least one event from a personal calendar of a second user.

23. The computer-implemented method of claim 1, wherein b) further comprises:

- b.1) retrieving at least one event from a group calendar.

24. The computer-implemented method of claim 1, wherein each of at least of a subset of the events has a location, and wherein b) comprises retrieving a plurality of events associated with the specified category and having a location corresponding to a specified location.

30

25. The computer-implemented method of claim 1, further comprising:

- displaying a list of categories identified as newly available.

26. The computer-implemented method of claim 1, further comprising:

- displaying a list of events identified as newly available.

27. The computer-implemented method of claim 1, further comprising:

- f) displaying an alert in connection with the selected event, at a predetermined time relative to the date of the event.

28. The computer-implemented method of claim 1, further comprising:

- f) outputting an audio alert in connection with the selected event, at a predetermined time relative to the date of the event.

29. The computer-implemented method of claim 1, further comprising:

- f) sending a communication in connection with the selected event, at a predetermined time relative to the date of the event.

30. A computer-implemented system for multi-layered online calendaring, comprising:

- a user input device for accepting user input specifying at least one of a plurality of categories, and for accepting user input selecting at least one of a plurality of displayed events;

wherein at least one category of the plurality of categories corresponds to a unique visual characteristic;

- an event retrieval module, coupled to the user input device, for retrieving a plurality of events associated with the specified category, each event having a date;

- a display module, coupled to the event retrieval module, for selectively displaying a date-delimited subset of the retrieved events associated with the specified category;

wherein the information for at least one event in the specified category is displayed using the unique visual characteristic corresponding to the specified category if the specified category corresponds to a unique visual characteristic; and

- a personal calendar storage device, coupled to the user input device, for adding a selected event to a computer-implemented personal calendar.

31. The computer-implemented system of claim 30, wherein the display module displays the computer-implemented personal calendar by overlaying information from at least two selected events from at least one specified category on a common display.

32. The computer-implemented system of claim 30, further comprising a purchasing module, coupled to the user input device, for processing a user-specified purchase associated with the selected event.

33. The computer-implemented system of claim 30, wherein:

- the user input device further accepts user input describing at least one personal event; and

the display module displays the computer-implemented personal calendar by overlaying the personal event onto the at least one selected event from the at least one specified category on a common display.

34. The computer-implemented system of claim 30, further comprising:

- an alert device, coupled to the personal calendar storage device, for displaying an alert in connection with the



31

selected event, at a predetermined time relative to the date of the event.

35. The computer-implemented system of claim 30, further comprising:

an alert device, coupled to the personal calendar storage device, for outputting an audio alert in connection with the selected event, at a predetermined time relative to the date of the event.

36. The computer-implemented system of claim 30, further comprising:

a group calendar publishing module, coupled to the personal calendar storage device, for selectively publishing the personal event to a group calendar storage device.

37. The computer-implemented system of claim 36, wherein:

the user input device accepts input specifying a group of users to have access to the group calendar; and

the group calendar storage device accepts input from a second user requesting information from the group calendar, and, responsive to the second user being specified in the group, selectively transmits the information from the group calendar to the second user.

38. A computer-implemented system for multi-layered online calendaring, comprising:

an events database, containing a plurality of events, each event associated with at least one category, each event having a date;

a personal category database, containing personal calendar information for a user;

wherein at least one category corresponding to the events database or the personal category database corresponds to a unique visual characteristic;

an application server, coupled to the events database and to the personal calendar database, for retrieving a plurality of events from the events database, the plurality of events being associated with a specified category, and for retrieving personal calendar information from the personal calendar database; and

a display device, coupled to the application server, for selectively displaying a date-delimited subset of the retrieved events associated with the specified category, the information for at least one event in the specified category is displayed using the unique visual characteristic corresponding to the specified category if the specified category corresponds to a unique visual characteristic.

39. The computer-implemented system of claim 38, further comprising a user input device coupled to the application service, wherein:

the application server, responsive to the user specifying an event from the displayed date-delimiting subset, adds the specified event to the personal calendar information for the user.

40. The computer-implemented system of claim 39, wherein the display device selectively displays a computer-implemented personal calendar by overlaying information from user-specified events.

41. A computer-implemented system for multi-layered online calendaring, comprising:

user input means, for accepting user input specifying at least one of a plurality of categories, and for accepting user input selecting at least one of a plurality of displayed events;

wherein at least one category of the plurality of categories corresponds to a unique visual characteristic;

32

event retrieval means, coupled to the user input means, for retrieving a plurality of events associated with the specified category, each event having a date;

display means, coupled to the event retrieval means, for selectively displaying a date-delimited subset of the retrieved events associated with the specified category, the information for at least one event in the specified category is displayed using the unique visual characteristic corresponding to the specified category if the specified category corresponds to a unique visual characteristic; and

personal calendar storage means, coupled to the user input means, for adding a selected event to a computer-implemented personal calendar.

42. The computer-implemented system of claim 41, wherein the display means displays the computer-implemented personal calendar by overlaying information from at least two selected events from at least one specified category on a common display.

43. The computer-implemented system of claim 41, further comprising purchasing means, coupled to the user input means, for processing a user-specified purchase associated with the selected event.

44. The computer-implemented system of claim 41, wherein:

the user input means further accepts user input describing at least one personal event; and

the display means displays the computer-implemented personal calendar by overlaying the personal event onto the at least one selected event from the at least one specified category on a common display.

45. The computer-implemented system of claim 41, further comprising:

alert means, coupled to the personal calendar storage means, for displaying an alert in connection with the selected event, at a predetermined time relative to the date of the event.

46. The computer-implemented system of claim 41, further comprising:

alert means, coupled to the personal calendar storage means, for outputting an audio alert in connection with the selected event, at a predetermined time relative to the date of the event.

47. The computer-implemented system of claim 41, further comprising:

group calendar publishing means, coupled to the personal calendar storage means, for selectively publishing the personal event to a group calendar storage means.

48. The computer-implemented system of claim 47, wherein:

the user input means accepts input specifying a group of users to have access to the group calendar; and

the group calendar storage means accepts input from a second user requesting information from the group calendar, and, responsive to the second user being specified in the group, selectively transmits the information from the group calendar to the second user.

49. A computer program product comprising a computer-usable medium having computer-readable code embodied therein for multi-layered online calendaring, comprising:

computer-readable program code devices configured to cause a computer to accept user input specifying at least one of a plurality of categories, and to accept user input selecting at least one of a plurality of displayed events;



33

wherein at least one category of the plurality of categories corresponds to a unique visual characteristic;

computer-readable program code devices configured to cause a computer to retrieve a plurality of events associated with the specified category, each event having a date;

computer-readable program code devices configured to cause a computer to selectively display a date-delimited subset of the retrieved events associated with the specified category, the information for at least one event in the specified category is displayed using the unique visual characteristic corresponding to the specified category if the specified category corresponds to a unique visual characteristic; and

computer-readable program code devices configured to cause a computer to add a selected event to a computer-implemented personal calendar.

50. The computer program product of claim 49, wherein the computer-readable program code devices configured to cause a computer to selectively display are configured to cause a computer to display the computer-implemented personal calendar by overlaying information from at least two selected events from at least one specified category on a common display.

51. The computer program product of claim 49, further comprising computer-readable program code devices configured to cause a computer to process a user-specified purchase associated with the selected event.

52. The computer program product of claim 49, wherein: the computer-readable program code devices configured to cause a computer to accept user input are configured to cause a computer to accept user input describing at least one personal event; and

the computer-readable program code devices configured to cause a computer to selectively display are config-

34

ured to cause computer to display the computer-implemented personal calendar by overlaying the personal event onto the at least one selected event from the at least one specified category on a common display.

53. The computer program product of claim 49, further comprising:

computer-readable program code devices configured to cause a computer to display an alert in connection with the selected event, at a predetermined time relative to the date of the event.

54. The computer program product of claim 49, further comprising:

computer-readable program code devices configured to cause a computer to output an audio alert in connection with the selected event, at a predetermined time relative to the date of the event.

55. The computer program product of claim 49, further comprising:

computer-readable program code devices configured to cause a computer to selectively publish the personal event to a group calendar storage device.

56. The computer program product of claim 55, wherein: the computer-readable program code devices configured to cause a computer to accept user input are configured to cause a computer to accept input specifying a group of users to have access to the group calendar; and

the group calendar storage device accepts input from a second user requesting information from the group calendar, and, responsive to the second user being specified in the group, selectively transmits the information from the group calendar to the second user.

\* \* \* \* \*

# USPTO TO PROVIDE ELECTRONIC ACCESS TO CITED U.S. PATENT REFERENCES WITH OFFICE ACTIONS AND CEASE SUPPLYING PAPER COPIES

In support of its 21<sup>st</sup> Century Strategic Plan goal of increased patent e-Government, beginning in June 2004, the United States Patent and Trademark Office (Office or USPTO) will begin the phase-in of its E-Patent Reference program and hence will: (1) **provide downloading capability of the U.S. patents and U.S. patent application publications cited in Office actions** via the E-Patent Reference feature of the Office's Patent Application Information Retrieval (PAIR) system; and (2) **cease mailing paper copies of U.S. patents and U.S. patent application publications with Office actions** (in applications and during reexamination proceedings) except for citations made during the international stage of an international application under the Patent Cooperation Treaty (PCT). In order to use the new E-Patent Reference feature applicants must: (1) obtain a digital certificate and software from the Office; (2) obtain a customer number from the Office; and (3) properly associate patent applications with the customer number. Alternatively, copies of all U.S. patents and patent application publications can be accessed without a digital certificate from the USPTO web site, from the USPTO Office of Public Records, and from commercial sources. The Office will continue the practice of supplying paper copies of foreign patent documents and non-patent literature with Office actions. Paper copies of cited references will continue to be provided by the USPTO for international applications during the international stage.

## Schedule

June 2004	TCs 1600, 1700, 2800 and 2900
July 2004	TCs 3600 and 3700
August 2004	TCs 2100 and 2600

All U.S. patents and U.S. patent application publications are available on the USPTO web site. However, a simple system for downloading the cited U.S. patents and patent application publications has been established for applicants, called the E-Patent Reference system. As E-Patent Reference and Private PAIR require participating applicants to have a customer number, retrieval software and a digital certificate, all applicants are strongly encouraged to contact the Patent Electronic Business Center to acquire these items. To be ready to use this system by June 1, 2004, contact the Patent EBC as soon as possible by phone at 866-217-9197 (toll-free), 703-305-3028 or 703-308-6845 or electronically via the Internet at [ebc@uspto.gov](mailto:ebc@uspto.gov).

## **Other Options**

The E-Patent Reference function requires the applicant to use the secure Private PAIR system, which establishes confidential communications with the applicant. Applicants using this facility must receive a digital certificate, as described above. Other options for obtaining patents which do not require the digital certificate include the USPTO's free Patents on the Web program (<http://www.uspto.gov/patft/index.html>). The USPTO's Office of Public Records also supplies copies of patents for a fee (<http://ebiz1.uspto.gov/oems25p/index.html>). Commercial sources also provide U.S. patents and patent application publications.

*For complete instructions see the Official Gazette Notice, USPTO TO PROVIDE ELECTRONIC ACCESS TO CITED U.S. PATENT REFERENCES WITH OFFICE ACTIONS AND CEASE SUPPLYING PAPER COPIES, on the USPTO web site.*

**NOTICE OF OFFICE PLAN TO CEASE SUPPLYING COPIES OF CITED U.S. PATENT  
REFERENCES WITH OFFICE ACTIONS, AND PILOT TO EVALUATE THE  
ALTERNATIVE OF PROVIDING ELECTRONIC ACCESS TO SUCH U.S. PATENT  
REFERENCES**

**Summary**

The United States Patent and Trademark Office (Office or USPTO) plans in the near future to: (1) cease mailing copies of U.S. patents and U.S. patent application publications (US patent references) with Office actions except for citations made during the international stage of an international application under the Patent Cooperation Treaty and those made during reexamination proceedings; and (2) provide electronic access to, with convenient downloading capability of, the US patent references cited in an Office action via the Office's private Patent Application Information Retrieval (PAIR) system which has a new feature called "E-Patent Reference." Before ceasing to provide copies of U.S. patent references with Office actions, the Office shall test the feasibility of the E-Patent Reference feature by conducting a two-month pilot project starting with Office actions mailed after December 1, 2003. The Office shall evaluate the pilot project and publish the results in a notice which will be posted on the Office's web site ([www.USPTO.gov](http://www.USPTO.gov)) and in the Patent Official Gazette (O.G.). In order to use the new E-Patent Reference feature during the pilot period, or when the Office ceases to send copies of U.S. patent references with Office actions, the applicant must: (1) obtain a digital certificate from the Office; (2) obtain a customer number from the Office, and (3) properly associate applications with the customer number. The pilot project does not involve or affect the current Office practice of supplying paper copies of foreign patent documents and non-patent literature with Office actions. Paper copies of references will continue to be provided by the USPTO for searches and written opinions prepared by the USPTO for international applications during the international stage and for reexamination proceedings.

**Description of Pilot Project to Provide Electronic Access to Cited U.S. Patent References**

On December 1, 2003, the Office will make available a new feature, E-Patent Reference, in the Office's private PAIR system, to allow more convenient downloading of U.S. patents and U.S. patent application publications. The new feature will allow an authorized user of private PAIR to download some or all of the U.S. patents and U.S. patent application publications cited by an examiner on form PTO-892 in Office actions, as well as U.S. patents and U.S. patent application publications submitted by applicants on form PTO/SB08 (1449) as part of an IDS. The retrieval of some or all of the documents may be performed in one downloading step with the documents encoded as Adobe Portable Document format (.pdf) files, which is an improvement over the current page-by-page retrieval capability from other USPTO systems.

## **Steps to Use the New E-Patent Reference Feature During the Pilot Project and Thereafter**

Access to private PAIR is required to utilize E-Patent Reference. If you don't already have access to private PAIR, the Office urges practitioners, and applicants not represented by a practitioner, to take advantage of the transition period to obtain a no-cost USPTO Public Key Infrastructure (PKI) digital certificate, obtain a USPTO customer number, associate all of their pending and new application filings with their customer number, install no-cost software (supplied by the Office) required to access private PAIR and E-Patent Reference feature, and make appropriate arrangements for Internet access. The full instructions for obtaining a PKI digital certificate are available at the Office's Electronic Business Center (EBC) web page at: <http://www.uspto.gov/ebc/downloads.html>. Note that a notarized signature will be required to obtain a digital certificate.

To get a Customer Number, download and complete the Customer Number Request form, PTO-SB125, at: <http://www.uspto.gov/web/forms/sb0125.pdf>. The completed form can then be transmitted by facsimile to the Electronic Business Center at (703) 308-2840, or mailed to the address on the form. If you are a registered attorney or patent agent, then your registration number must be associated with your customer number. This is accomplished by adding your registration number to the Customer Number Request form. A description of associating a customer number with an application is described at the EBC web page at: [http://www.uspto.gov/ebc/registration\\_pair.html](http://www.uspto.gov/ebc/registration_pair.html).

The E-Patent Reference feature will be accessed using a new button on the private PAIR screen. Ordinarily all of the cited U.S. patent and U.S. patent application publication references will be available over the Internet using the Office's new E-Patent Reference feature. The size of the references to be downloaded will be displayed by E-Patent Reference so the download time can be estimated. Applicants and registered practitioners can select to download all of the references or any combination of cited references. Selected references will be downloaded as complete documents as Adobe Portable Document Format (.pdf) files. For a limited period of time, the USPTO will include a copy of this notice with Office actions to encourage applicants to use this new feature and, if needed, to take the steps outlined above in order to be able to utilize this new feature during the pilot and thereafter.

During the two-month pilot, the Office will evaluate the stability and capacity of the E-Patent Reference feature to reliably provide electronic access to cited U.S. patent and U.S. patent application publication references. While copies of U.S. patent and U.S. patent application publication references cited by examiners will continue to be mailed with Office actions during the pilot project, applicants are encouraged to use the private PAIR and the E-Patent Reference feature to electronically access and download cited U.S. patent and U.S. patent application publication references so the Office will be able to objectively evaluate its performance. The public is encouraged to submit comments to the Office on the usability and performance of the E-Patent Reference feature during the pilot. Further, during the pilot period registered practitioners, and applicants not represented by a practitioner, are encouraged to experiment with the feature, develop a proficiency in using the feature, and establish new internal processes for using the new access to the cited U.S. patents and U.S. patent application publications to prepare for the anticipated cessation of the current Office practice of supplying copies of such cited

references. The Office plans to continue to provide access to the E-Patent Reference feature during its evaluation of the pilot.

### Comments

Comments concerning the E-Patent Reference feature should be in writing and directed to the Electronic Business Center (EBC) at the USPTO by electronic mail at [eReference@uspto.gov](mailto:eReference@uspto.gov) or by facsimile to (703) 308-2840. Comments will be posted and made available for public inspection. To ensure that comments are considered in the evaluation of the pilot project, comments should be submitted in writing by January 15, 2004.

Comments with respect to specific applications should be sent to the Technology Centers' customer service centers. Comments concerning digital certificates, customer numbers, and associating customer numbers with applications should be sent to the Electronic Business Center (EBC) at the USPTO by facsimile at (703) 308-2840 or by e-mail at [EBC@uspto.gov](mailto:EBC@uspto.gov).

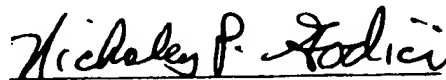
### Implementation after Pilot

After the pilot, its evaluation, and publication of a subsequent notice as indicated above, the Office expects to implement its plan to cease mailing paper copies of U.S. patent references cited during examination of non provisional applications on or after February 2, 2004; although copies of cited foreign patent documents, as well as non-patent literature, will still be mailed to the applicant until such time as substantially all applications have been scanned into IFW.

### For Further Information Contact

Technical information on the operation of the IFW system can be found on the USPTO website at <http://www.uspto.gov/web/patents/ifw/index.html>. Comments concerning the E-Patent Reference feature and questions concerning the operation of the PAIR system should be directed to the EBC at the USPTO at (866) 217-9197. The EBC may also be contacted by facsimile at (703) 308-2840 or by e-mail at [EBC@uspto.gov](mailto:EBC@uspto.gov).

Date. 12/1/03

  
Nicholas P. Godici  
Commissioner for Patents

U. S. DEPARTMENT OF COMMERCE

COMMISSIONER FOR PATENTS

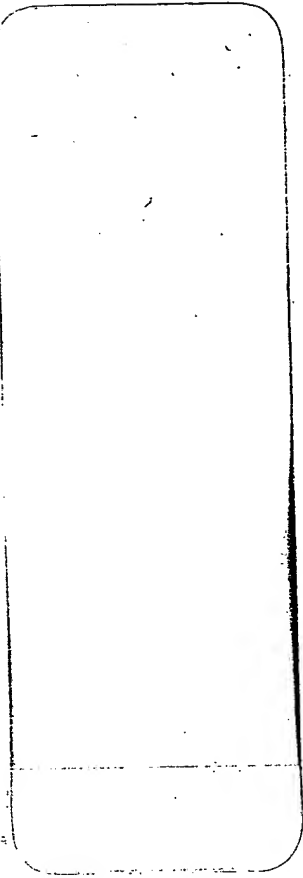
200-450-4450

AT 1000 ANDRIA WA 22313-1450

RECEIVED BY MAIL RETURN IN TEN DAYS

OFFICE OF ALI 101 STREET

AN EQUAL OPPORTUNITY



 **MOVED, LEFT  
NO ADDRESS**

**RECEIVED**

**FEB 24 2005**

**Technology Center 2100**